5-2013

Automating Large-Scale Simulation Calibration to Real-World Sensor Data

Richard Everett Edwards
redwar15@utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Richard Everett Edwards entitled "Automating Large-Scale Simulation Calibration to Real-World Sensor Data." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Science.

Lynne E. Parker, Major Professor

We have read this dissertation and recommend its acceptance:

Michael Berry, Hamparsum Bozdogan, Husheng Li, Joshua New

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Automating Large-Scale Simulation Calibration to Real-World Sensor Data

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Richard Everett Edwards

May 2013

Many people have contributed to my inspiration and desire to complete this work, despite their occasional nagging question, "When are you going to finish exactly?". I dedicate this dissertation to the my family, friends, and the following people:

Grandfather Russell Busbee – In addition to working his way up from digging ditches to an accountant at DuPont, he made sure my Aunt Dell, Uncle Rusty, and my mother had the opportunity and the ability to obtain college degrees.

Grandfather Richard E. Edwards Sr. – He passed November 2012 from pancreatic cancer at age 81. He always encouraged his grandchildren to pursue math, science, and engineering. In addition, he dedicated his entire life's attention to his family.

Mother, Nancy B. Powell – She forced me to study as a child, despite my desires to play and pursue other endeavors, and she never allowed me to settle for anything less than what my abilities should achieve. In addition, she supported me throughout all studies with text book expenses, living expenses, and more.

Step Father, Clay Gehrke – He was always there to help me move. Clay helped move me in and out of every dorm room and apartment without complaint. This includes him helping me move all the way from Houston to Knoxville.

Wife, Xia (Sophia) Huang – While Sophia was very pushy about my graduate studies, she always made sure I had excellent food and rest. I have greatly enjoyed her home cooked lunches, and I look forward to her continued support.

Father, Richard E. Edwards Jr. – My father paid all my tuition at the University of Texas, and allowed me to spend extra time there to explore undergraduate research. My explorations into research as an undergraduate student ultimately lead me here.

Acknowledgements

I would like to thank Dr. Joshua New for funding and assisting all research presented within this dissertation. In addition, I would like to thank Dr. Lynne E. Parker for providing guidance throughout my graduate studies and research as well as providing opportunities to grow and expand my knowledge within the machine learning domain. Additionally, I would like to thank Dr. Hamparsum Bozdoghan for evolving my understanding about model complexity and model selection overall, it has greatly changed how I approach most learning problems. Lastly, I would like to thank all my friends and family for their continuous support throughout my graduate studies.

Abstract

Many key decisions and design policies are made using sophisticated computer simulations. However, these sophisticated computer simulations have several major problems. The two main issues are 1) gaps between the simulation model and the actual structure, and 2) limitations of the modeling engine's capabilities. This dissertation's goal is to address these simulation deficiencies by presenting a general automated process for tuning simulation inputs such that simulation output matches real world measured data. The automated process involves the following key components – 1) Identify a model that accurately estimates the real world simulation calibration target from measured sensor data; 2) Identify the key real world measurements that best estimate the simulation calibration target; 3) Construct a mapping from the most useful real world measurements to actual simulation outputs; 4) Build fast and effective simulation approximation models that predict simulation output using simulation input; 5) Build a relational model that captures inter variable dependencies between simulation inputs and outputs; and finally 6) Use the relational model to estimate the simulation input variables from the mapped sensor data, and use either the simulation model or approximate simulation model to fine tune input simulation parameter estimates towards the calibration system.

The work in this dissertation individually validates and completes five out of the six calibration components with respect to the residential energy domain. Step 1 is satisfied by identifying the best model for predicting next hour residential electrical consumption, the calibration target. Step 2 is completed by identifying

the most important sensors for predicting residential electrical consumption, the real world measurements. While step 3 is completed by domain experts, step 4 is addressed by using techniques from the Big Data machine learning domain to build approximations for the EnergyPlus (E+) simulator. Step 5's solution leverages the same Big Data machine learning techniques to build a relational model that describes how the simulator's variables are probabilistically related. Finally, step 6 is partially demonstrated by using the relational model to estimate simulation parameters for E+ simulations with known ground truth simulation inputs.

Contents

List of Tables

List of Figures

Introduction

Many key decisions and design policies are made using sophisticated computer simulations. For example, building engineers use whole energy building simulations to estimate building electrical consumption. They use these estimates to measure how policy changes in building materials, building design, occupancy behavior, and much more influences energy efficiency and overall savings. However, these sophisticated computer simulations have several major problems. The two main issues are 1) gaps between the simulation model and the actual structure, and 2) limitations of the modeling engine's capabilities

Gaps between as-modeled and as-built structures come in many sources and ultimately relate to the addage "garbage in, garbage out" with the fault lying within the inaccurate input file rather than the simulation engine itself. For example, in the building energy domain, Infiltration, the rate at which air and the energy in it flows through the building envelope (typically measured in cubic feet per minute per square ft), is not currently able to be cheaply tested despite being one of the most important factors for building energy efficiency. Blower-door tests can determine infiltration rate at a given pressure (usu. 50 Pascals) but is a 1-time measurement that, in reality, experiences significant variances as a function of temperature, wind speed, wind direction, etc. As such, infiltration is often one of the first variables most energy modeling experts use to manually align a simulation model with actual data.

Another gap is overall duty cycle and overall usage. For example, the building energy domain models overall building usage with a yearly operation schedule file,

which includes number of occupants, times of occupancy, HVAC setpoints, device operations schedule, and many other factors. For many of these, cost-effective sensors simply do not exist or are not typically deployed in a building (especially data provided in a way to be easily leveraged by energy modelers). In many cases, estimates of occupancy schedules and relatively static setpoint temperatures are estimated and then used later to "true-up" the simulation to match whole-building data without regard to the accuracy of the actual HVAC thermostat setpoints.

The third cause can be attributed to inaccurate information about model components. Most modelers typically use the advertised material properties (i.e., product labels) or standard values, which are typically published in reference manuals. However, laboratory testing has shown that material properties are a major contributor to model variance, which means inaccurate values have significant consequences. Occasionally, the manufacturer labels are more accurate than the reference manuals. However, laboratory-controlled testing of specific materials has shown significant variance in materials even from a single manufacturer. In such cases, modelers have very little reliable data to determine the precise values necessary for creating an accurate simulation model.

The final common gap is modelers typically use the original design specifications, which may differ from the actual built structure. Physical structures tend to differ slightly from the original design specification, which is attributed to either variation in the final construction process or the actual environmental conditions producing different results than the original design assumptions.

Limitations of engine modeling capabilities is a more well-understood and active area of involvement with funded development teams or active communities behind the most popular simulation engines and tools. Unlike making a one time use simulation model accurate, there is sufficient market incentive and policy impact to be found in developing a capable software engine that can be either sold or used to develop environmental policies, energy saving policies, and many more. However, there are a few primary factors that may be considered in relation to inaccuracies

of simulation engines. First, most simulation engines are engineering models that attempt to model, through time and with some degree of fidelity, the underlying physics model. As is necessary from such an approach, engineering algorithms are by necessity an approximation of reality (e.g., using 1D heat transfer equations over 3D heat transfer equations). Statistical models show some promise, but are more useful in normative-based models for policy decisions rather than product-level or system-level modifications. Second, there is a lag between the development of new, innovative technologies and the capabilities of accurately modeling it within a simulation engine. Only the most active simulation engine development teams are able to keep up with or foresee the need to model new products, components, systems, or additional influences before they achieve a significant market share or an accepted impact within the community. In addition, as the codebase grows, the challenge of maintaining a software architecture that can accommodate new integrated technologies (which may impact several simulation components) also grows. Third, many simulation engines are single-threaded with only recent attempts to leverage multi-core computational hardware. Given the significantly different multi-threaded software development paradigms, there will be substantial challenges in being able to scale additional simulation capabilities without an increase in runtime for models that use those capabilities. Fourth, the computer itself is an approximation engine and is fundamentally limited with respect to the accuracy that it can provide in a unit of time for a given algorithm. For most simulation engine developments, the focus has been a reactive process of building something that is sufficient to meet some small fraction of the long list of needs expressed by the users.

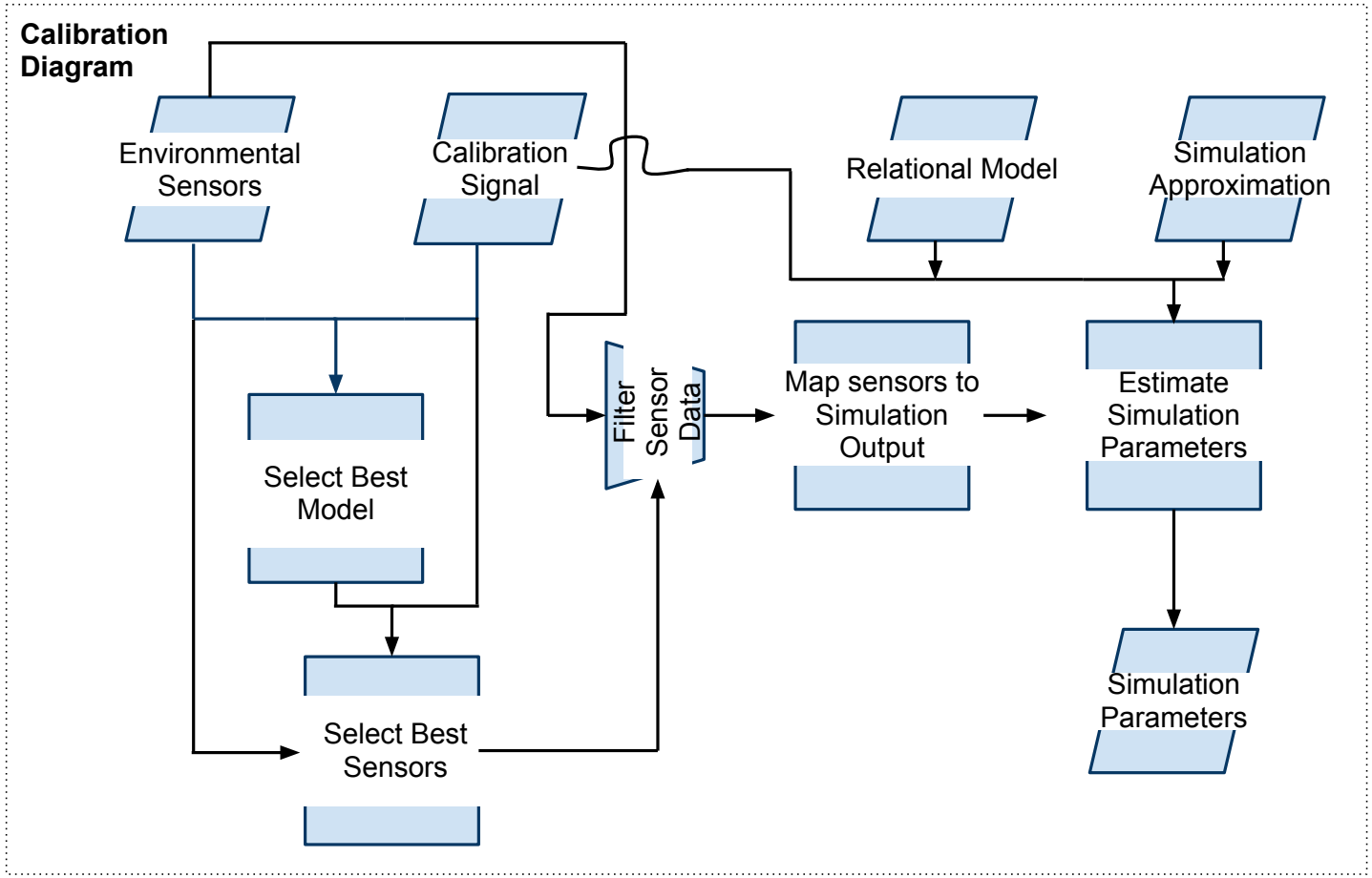


Figure 1: Automated simulation calibration process diagram

This dissertation's goal is to address these simulation deficiencies by presenting a general automated process for tuning simulation inputs such that simulation output matches real world measured data. The automated process, depicted in Figure 1, involves the following key components – 1) Identify a model that accurately estimates the real world simulation calibration target from measured sensor data (Chapter 3); 2) Identify the key real world measurements that best estimate the simulation calibration target (Chapter 4)); 3) Construct a mapping from the most useful real world measurements to actual simulation outputs; 4) Build fast and effective simulation approximation models that predict simulation output using simulation input* (Chapter 5); 5) Build a relational model that captures inter variable dependencies between simulation inputs and outputs (Chapter 6); and finally 6) Use the relational model to estimate the simulation input variables from the mapped sensor data, and use either the simulation model or approximate simulation model to fine tune input simulation parameter estimates towards the calibration system. It should be noted that the fine tuning phase must respect structural dependencies within the relational model when augmenting input parameter estimates, (i.e., the relational model imposes constrained optimization over the simulation inputs).

The remainder of this chapter presents high level comments and highlights key components within the automated simulation calibration process (Approach Overview Section); Contribution section presents all contributions found within this dissertation. Finally, the Dissertation Overview section presents a high-level outline for all the chapters within the dissertation.

Approach Overview

In order to validate and explore the automated simulation calibration system's capabilities, this dissertation focuses on the whole building simulation domain. While

* Only required if the overall simulation engine is extremely slow, making it difficult to run many simulations

this dissertation focuses on the building simulation domain, the entire process and individual techniques are application independent. This means all the methods and algorithms within this dissertation are able to calibrate any simulation software, provided all necessary simulation and environmental data is available.

EnergyPlus (E+) is a sufficiently complicated large-scale whole-building simulation model (Section 2.1 provides more details) with available simulation data. In addition, there exist real world sensor data for the simulated building (i.e., Wolf Creek sensor data (Section 2.2.2)). However, this research does not address the third component within the automatic calibration process. The mapping from real world sensor data to simulation outputs was hand crafted by domain experts, which means this component is not automated. In addition, the domain knowledge experts are providing a key learning bias required to facilitate calibration, by ultimately selecting the most important simulation outputs and sensors used for calibration, rather than completely using the recommended sensors produced by step 2.

Given this information, this research addresses step 1 by exploring many different regression methods using the Campbell Creek data set (Secton 2.2.1). Using these different regression methodologies, we can isolate the best model for estimating the calibration target from the available sensor data. In this instance, our calibration signal is the overall building's future electrical consumption, a fairly difficult response variable to estimate. The tools used to complete step 1 are packaged and readily usable for estimating other calibration signals from any sensor data set.

Step 2 is addressed by exploring feature selection methods that are easily combined with the best predictor from step 1. In particular, our results indicate that the Information Complexity measure and our novel voting method facilitates selecting the best sensors.

Needing to address step 4 is purely dependent upon the overall simulation engine's performance and execution time. As mentioned in Section 2.1, E+ simulations require approximately 2-3 minutes each on a standard desktop computer. While this execution time may seem insignificant, calibrating simulation inputs using

such a slow model is very cumbersome, especially if the total number of required evaluations approach several thousand. Therefore, this dissertation explores the Big Data community's algorithmic methods and adopts a scaleable learning algorithm for approximating E+ simulations. In addition, this dissertation modifies an existing gradient method using a standard stochastic-batch hybrid update method, which optimizes sequential gradient descent learning performance.

Step 5 is addressed by combining graphical model structural learning with the same scaleable learning algorithm from the Big Data community. We also explore the traditional direct learning approach, which assumes no priors, as well as a Bayesian structural learning approach, which assumes a Wishart prior.

Using these components it is possible to complete the actual calibration process, step 6, with real world sensor data. However, we have yet to experimentally test the calibration process using real world sensor data. Rather all components within the system were validated individually using either known or approximate ground truth information. We are currently working on fully validating the system and plan to incorporate E+ simulation results for the real building's estimated parameters compared against the measured ground truth sensor data.

Contributions

The following list contains my Building Spaces contributions:

- Best predictor for hourly residential electrical consumption (Chapter 3)
- Best sensors for predicting electrical consumption (Chapter 4)
- First general purpose large-scale E+ residential approximation (Chapter 5)

The following list contains my Computer Science contributions:

- A novel feature selection method, which uses the estimated ICOMP distribution over the features to select the best ones via voting (Chapter 4)

- Adapting Bayesian and Direct regression structure learning to large-scale datasets, via Alternating Direction Method of Multipliers (Chapter 6)

- General large-scale automated computer simulation calibration process (Figure 1)

These contributions have individual benefits, as well as facilitate constructing the overall general automated calibration process. Knowing the best predictor for hourly residential electrical consumption provides building engineers with an excellent starting point when modeling a new residential building. Ideally, the same model will be the best for all buildings, but the results within this dissertation illustrate empirical model evaluation, which can guide the building engineer's model selection process. However, selecting the best sensors for predicting electrical consumption is greatly dependent upon the underlying predictive model's capabilities, which means identifying the best model is essential for selecting the most important sensors. In addition, selecting the most important sensors helps reduce installation and planning costs for future building instrumentation.

A general purpose large-scale E+ approximation that runs in a few seconds allows building engineers to manually calibrate simulations much more effectively. In addition, it alleviates computation time consumed by automatic calibration methods that try to use the simulator to facilitate calibration by directly executing simulations. However, large-scale approximation models require large simulation data sets, which exceed standard regression methods' capabilities. Solving this difficulty by utilizing a regression method that fully scales to any regression problem, directly solves most challenges associated with learning linear inter-variable dependencies, or inter-variable structural dependencies, over the simulation variables.

The E+ inter-variable dependency model allows building engineers to statistically infer building parameters automatically from data. However, the overall linear inter-variable dependency method is completely general and applicable to any large scale structural learning problem.

Finally, combining all contributions together allows full simulation calibration. A simulation calibration engineer can select the best predictive model and use that model along with the feature selection methods in this dissertation to select the most important environmental sensors. Given the best sensors, the engineer can select corresponding simulation outputs that map well with the best sensors. Using the manually derived sensor to simulation output mapping, the engineer can use the faster approximations to either manually or automatically calibrate the simulation. The automatic calibration process would combine any standard optimization method with the fast approximations. Alternatively, the engineer can use the relational model to infer parameter estimates automatically by using reference sensor data. These estimates can serve as initial calibration starting points for manual calibration or as the final calibration parameters by fine tuning them with the simulation approximation.

Dissertation Overview

The remainder of this dissertation is structured as follows:

- Chapter 1 presents related work on simulation calibration methods, simulation approximation methods, modeling electrical consumption using sensors, and previous studies on sensor selection.
- Chapter 2 presents background material on the E+ simulator and provides a summary for all data sets used in this work.
- Chapter 3 compares traditional modeling against sensor-based modeling. In addition, it presents several techniques that were explored for performing sensor-based energy modeling and their corresponding prediction results on the Campbell Creek data set.

- Chapter 4 analyzes different feature selection methodologies and tests how well wrapper based feature selection performs at selecting the best sensors for predicting electrical consumption.

- Chapter 5 explores adjusting the FFNN learning process by using a stochastic-batch hybrid updating process, which allows it to approximate E+ simulations. In addition, it presents using Lasso regression with Alternating Direction Method of Multipliers (ADMM) to quickly fit linear models.

- Chapter 6 analyzes previous structural learning methods with respect to scalability. The chapter illustrates that regression based structure learning is extremely scaleable and presents two methods that scale well to the E+ data set by leveraging the Lasso regression method from Chapter 5.

- The conclusion summarizes how each individual chapter 3-6 addresses its corresponding step within the automated calibration process. In addition, it presents all contributions and future directions for exploring this research topic and application.

Chapter 1

Related Work

1.1 Simulation Calibration

There are many different simulation calibration approaches through out the literature. However, all approaches can be categorized into two areas: manual procedural calibration methodologies and semi-automated statistical calibration procedures. This first category generally involves constructing a manual audit process for incorporating environmental information into the engineer's simulation parameter calibration process. Raftery et al. (2011) provides an excellent example. This work integrates measurements, and additional periodic audits into the calibration process, which creates a fairly reliable manual procedure. The process is more reliable because the engineer now incorporates external temporal conditions that induce measurement variations into the simulation parameter adjustments. Generally, the calibration process may only use a single environmental snap shot for the simulated target. Yoon and Lee (1999) and Pedrini et al. (2002) present older manual calibration methodologies, which are similar to Raftery et al. (2011). All manual calibration processes at their core identify the most important tuning parameters via simulation sensitivity analysis. This means the engineer runs simulations with different parameter settings and estimates which parameters provide the largest

change in the simulation's overall behavior with respect to the target. Westphal and Lamberts (2005) presents an example sensitivity analysis study for calibrating whole-building energy simulations. While these manual approaches are effective, the entire process must be repeated per building. In addition, it is a time consuming process, which means a large amount of engineering time is invested in calibrating a single simulation.

Alternatively, other engineers approach the problem analytically using statistical models. The statistical model approach involves engineers or domain experts selecting the most important simulation parameters, model outputs, environmental measurements for calibration, and mappings between simulation outputs and environmental measurements, which results in a semi-automated calibration process. For example, Tian and Choudhary (2012) builds small scale simulation approximations or surrogates, and uses these surrogates to optimize the simulation parameters with respect to known target data. The surrogates are not necessarily required, as shown by the Zeng et al. (2013), which directly calibrates parameters using the actual simulation model. However, the simulation execution time in Zeng et al. (2013) was two minutes per simulation and the authors ran 13,000 simulations. This means that their calibration process is not realistic for general purpose use without the parallel computation and computing resources used in their work. The overall execution time in serial is 18 days. While some methods are slower than others, these semi-automated statistical approaches are transferable and repeatable across multiple domains and simulation environments, provided experts or engineers select the most relevant criteria. This dissertation partially addresses this semi-automated nature, by using feature selection to select the most important sensors with respect to the overall most important calibration target (Chapter 4). In addition, it explores constructing large-scale surrogate models that can facilitate calibration, and reduce the calibration runtime to a range manageable by a more general user base (Chapter 5). Lastly, the overall computation burden is reduced further by providing a single compact model

that may be used either standalone or in conjunction with the approximation models to infer simulation parameters directly from reference data (Chapter 6).

1.2 Surrogate Modeling

Surrogate generation or meta-modeling within the building domain typically uses a few classical statistical techniques – Ordinary Least Squares (OLS) linear regression, Multivariate Adaptive Regression Splines (MARS), Kriging*, and Radial Basis Functions (RBF). Each technique has its own strengths and weaknesses, which are documented throughout the literature Jin et al. (2001). The first two are fairly popular because they also facilitate sensitivity analysis Storlie et al. (2009); Helton et al. (2006), which allows researchers to determine the key input variables for particular outputs. Fitting a surrogate model that uses the key inputs allows researchers to calibrate simulations around these key inputs by solving an inverse optimization problem using actual measured data Tian and Choudhary (2012), which is more efficient than directly solving the inverse optimization problem with the exact software simulation.

Overall calibration quality is dependent upon the surrogate model's estimation accuracy. The work in Tian and Choudhary (2012) illustrates that small scale (16 inputs and 1 output) E+ surrogates are able to produce accurate distribution estimates over parameter settings for buildings based on actual measured data. In addition, Tian and Choudhary (2012) used linear regression and MARS to generate surrogate models and noted the need to explore other surrogate model options. However, the work focuses on large macro-scale building stock parameter estimation, which reduces the overall surrogate model's size and complexity. Our work, on the other hand, focuses on producing surrogate models for large scale E+ residential simulations (156 inputs and 80 to 90 output variables) for individual buildings.

* This method is also referred to as Gaussian Process Regression in the literature.

The scale difference produces significant scalability issues with fitting the MARS, Kriging, and RBF surrogates. In particular, the computational time and memory requirements quickly become intractable as the available model training data increases. This scalability issue motivated us to explore machine learning methods from state-of-the-art data mining methodologies for Big Data. This exploration ultimately lead us to select FFNN and Lasso regression using Alternating Direction of Method of Multipliers Boyd et al. (2011), which are discussed more in Sections 5.1.1 and 5.1.3. These methods allow us to produce large scale surrogate models and determine their overall effectiveness at producing actual E+ simulation outputs.

1.3 Sensor Modeling

Many researchers have explored machine learning alternatives for modeling electrical consumption, both within commercial buildings and residential buildings. However, a majority of the studies have focused on commercial buildings. A notable study that used commercial building data is the Building Energy Predictor Shootout hosted by ASHRAE. The competition called for participants to predict hourly whole building electrical (wbe) consumption for an unknown building using environmental sensors and user-defined domain knowledge. The competition provided 150 competitors with data from September 1, 1989 until December 31, 1989 as training data, as well as testing data that had the target variables removed. Six winners were selected from the submitted predictions Kreider and Haberl (1994).

The overall winner, MacKay et al. (1994), used a Feed Forward Neural Network (FFNN) with Auto Relevance Detection (ARD). The author was not sure which inputs or variables were most beneficial for predicting the specified targets. Therefore, the author devised a method for exploring a wide variety of different inputs that would minimize the error caused by irrelevant inputs. This Auto Relevance Detection process drives the weights for irrelevant inputs toward zero and prevents the weights for other inputs from growing too large or overpowering the solution. This is achieved

by reformulating weight regularization to obey a probabilistic model, where all parameters follow prior distributions and the weights are inferred using Bayesian inference. The results presented from this prior work provide strong incentive for exploring how effective FFNNs are at predicting future residential electrical consumption. Our use of this method is discussed in more detail in Section 3.4.

Another winner used Piecewise Linear Regression Iijima et al. (1994). The authors created three different linear functions for predicting wbe. The first model is dedicated to workdays, the second is dedicated to weekends, and the third is dedicated to modeling holidays. These models were combined using the provided temporal information: day, month, year, and hour. However, the method used in this work requires explicit temporal domain knowledge about the particular application area. Given that we lack such temporal domain knowledge for residential domains, we decided to explore an automated Piecewise Linear Regression process. We apply Hierarchical Mixture of Experts (HME) with Linear Regression, because it uses the training data to automatically build and integrate multiple linear models. Section 3.3 briefly describes Linear Regression, and Section 3.7 discusses HME with Linear Regression in greater detail.

Feuston and Thurtell (1994) used an ensemble of FFNNs, which involved training multiple FFNNs and combining them by averaging their predictions. The predictions for each FFNN were equally weighted and the networks were trained using the same training data, and possibly different initializations. This method assumes that all FFNN responses are equally important. This assumption can harm accuracy, especially if a majority of the FFNNs learn the same errors, and only a few networks learn to correct those errors. Therefore, we decided to explore a more balanced and general method for mixing multiple FFNNs. The HME approach, which we previously mentioned, combined with FFNN Experts, accomplishes the same task, except the predictions are combined based on the likelihood that each network produces the correct prediction (Section 3.7).

A more recent wbe prediction study with commercial buildings uses Support Vector Machines (SVM) to predict monthly consumption Dong et al. (2005). SVMs are built on the principle that minimizing structural risk produces a general model. In addition, SVMs have a proven upper bound on the error rate for classification problems Vapnik (1999). While we do not know of a proven upper bound for regression problems, minimizing structural risk can still produce general models. The results from this prior work and the known benefits from SVMs lead us to the application of Support Vector Regression (SVR), which is SVM adapted for Regression (Section 3.5). These prior results also encouraged us to explore an SVM variant, called Least Squares Support Vector Machine (LS-SVM) (Section 3.6).

Karatasou et al. (2006) builds upon the success found with FFNN and explores selecting the most important inputs and network structure for the Building Energy Predictor Shootout data. In addition, the work explores another commercial building data set. The authors present impressive results on both buildings and out-performed the Shootout winner. However, the authors provide little discussion about what allowed them to obtain better performance or the key differences between other FFNN techniques. The results found within this study provide further incentive to explore the application of FFNN to predicting residential electrical consumption.

Another recent work, by Li et al. (2011), presents results for the Energy Predictor Shootout that are better than the overall winner as well. This approach uses an Adaptive Neuro Fuzzy Inference System (ANFIS), which deviates greatly from the previously published FFNN works. This method combines partitioning rules from Fuzzy Systems with the properties of FFNNs, which is similar to Fuzzy C-Means (FCM) with FFNN. However, the authors in this work fully use the Fuzzy Systems by using multiple partitioning functions, while the FCM with FFNN in our work uses a single partitioning function. Section 3.8 provides a more detailed description about FCM with FFNN.

These studies on commercial buildings provide insight into successful techniques, many of which have inspired several of the techniques we explore in this research.

However, how successful are these techniques on residential buildings? The studies that involve residential buildings are generally conducted with monthly information collected from utility companies. This means that most residential studies do not provide hourly predictions, which is fairly different from our focus on predicting hourly wbe consumption. For instance, Kolter and Ferreira Jr (2011) focuses on modeling commercial and residential buildings, but all the whole building energy (wbe) measurements are only at a monthly resolution for all buildings. This restriction is created by the fact that utility companies measure residential electrical consumption at monthly intervals, while commercial electrical consumption is measured hourly.

Our research makes use of a new residential data set, called the Campbell Creek data set, which gives us a unique opportunity to predict next hour wbe consumption for residential homes. The Campbell Creek data set contains approximately 140 different sensor measurements collected every 15 minutes. We explain this data set in more detail in Section 2.2.1. This data set provides a vast quantity of inputs that far surpasses the amount of information used in the previous commercial and residential building studies. For example, the Great Energy Prediction Shootout data set contains only five measurements per hour. This means we are able to test existing techniques that were proven on previous smaller data sets, and introduce new techniques that have not previously been applied to this field.

Chapter 2

Preliminaries

This chapter outlines all preliminary material required to assimilate this dissertation. Section 2.1 provides a detailed description of the simulator used as the calibration test platform. In addition, Section 2.2 describes all data sets used throughout this dissertation. Section 2.2.1 describes the data sets used in Chapters 3 and 4, and Sections 2.2.2 and 2.2.3 describes the data sets used in Chapters 5 and 6. Finally, Section 2.3 illustrates the difficulties associated with scaling the better techniques in Chapter 3 to large data sets or Big Data.

2.1 E+

E+ is currently DOE's flagship whole-building energy simulation engine developed with active involvement by many participating individuals and organizations since 1996, with roots dating back to DOE-2 and Building Loads Analysis and System Thermodynamics (BLAST) from the late 1970s. DOE has trademarked the EnergyPlus name while copyright and intellectual property for E+ is held by Lawrence Berkeley National Laboratory. DOE has recently provided an open source license agreement in addition to the executable distribution and cost-based commercial source license DOE (2012b). A branch of the official E+ development trunk has also been posted on SourceForge Peter Ellis (2012). E+ consists of ∼600k lines of

Fortran code, but utilizes a much more extensible, modular architecture than DOE-2 to perform the energy analysis and thermal load simulation for a building. The extensive capabilities of E+ are beyond the scope of this dissertation; the interested reader is referred to existing resources DOE (2012a,c,d) for further information. However, the computational costs of these capabilities has resulted in annual building simulations that, depending on the complexity of the building information, often requires 5+ minutes (10x-100x slower than DOE-2 Hong et al. (2008)) of wall-clock time to complete; reducing the runtime of E+ is the top priority of the development team with EnergyPlus 7.0 being 25%-40% faster than EnergyPlus 6.0 DOE (2011).

The simulation engine uses a building specification file and a schedule file. The building file specifies all the building's physical properties, while the schedule describes when components within the environment will turn on and off. The schedule should represent the expected occupancy behavior for the particular building. For instance, the residential dataset's (Section 2.2.1 and 2.2.2) occupancy behavior is based on a standard family. While the average may not always be the best occupancy, it provides an overall expected view of a building under standard operating conditions.

The final simulation input component is weather data. E+ depends upon actual weather data, or estimated weather data, to drive its internal models. Using this component and the previous two components, E+ can simulate a building for an entire year. However, the simulation must be conducted on a single core and in some instances can take about eight minutes per simulation. The slow turn around for simulations makes the tuning process rather cumbersome and time consuming, especially when a building specification file may contain 1000 to 3000 tunable parameters. Therefore, being able to approximate and tune the whole building simulation model is very valuable to the Energy community.

2.2 Data Sets

2.2.1 Campbell Creek

The new residential data set used in our research, called the Campbell Creek data set, is a rich and unique data set. This data set was collected from three different homes located in west Knox County, Tennessee. These Campbell Creek homes are leased and operated by Tennessee Valley Authority (TVA) as part of a study testing energy efficient materials and their savings Christian et al. (2010). The first house in this study, called House 1, is a standard two-story residential home. However, the second and third houses, called House 2 and House 3 respectively, were modified to decrease energy consumption. House 2 uses the same construction materials as House 1, but was retrofitted with more energy efficient appliances, water heater, and HVAC. House 3 was built using construction techniques and materials designed to help reduce energy consumption. In addition, House 3 has a set of photovoltaics for generating electricity and a solar thermal water heater.

In this dataset each house has approximately 140 different sensors that collect data every 15 minutes. Each house is also outfitted with automated controls that manage the opening/closing of the refrigerator door, using the oven, running clothes washer and dryer, as well as shower usage. These automatic controls achieve an occupancy pattern that is consistent with typical energy usage patterns of American households, as determined by a Building America study Hendron et al. (2010). The simulated occupancy provides stable behavioral patterns across all three homes, making device usage within the data set consistent across test environments. This means the data set is free from behavioral factors, making it easier to compare results for different houses. Note that this data set provides a vast quantity of inputs that far surpasses the amount of information used in the previous commercial and residential building studies.

Removing the dynamic human behavior variable is clearly advantageous for making better predictions. However, these three homes were used to conduct numerous experiments throughout the data collection process. This means there were equipment substitutions, thermostat set point changes, prototype equipment tests, and much more. Therefore, the data sets still exhibits rich dynamic behavior, unless the data collected from these experiments is removed or treated as special cases.

2.2.2 Wolf Creek

There are four houses in the Wolf Creek data set and each one is equipped with approximate 250 sensors. All sensor measurements are at a 15 minute resolution just like the Campbell Creek homes. However, unlike the Campbell Creek houses, these houses were built using the latest building materials and energy savings technologies. The objective was to measure how much electricity could be saved, even when using materials that are cost prohibitive to the market adoption.

While the buildings in this data set are more advance than the Campbell Creek buildings, and most likely present interesting consumption and behavior patterns, these buildings serve a different purpose. Building engineers have been working hard to manually tune building models for the Wolf Creek homes. In particular, Wolf Creek 1 (WC1) is the reference building used for the E+ simulation data that is discussed in the following section. Using the sensor data from WC1, we can estimate how well our calibration process is able to match the manual tuning endeavors, and estimate how much time is saved with respect to human hours spent working on adjusting simulation parameters.

2.2.3 E+ Simulations

The Autotune project* Sanyal et al. (2012) has generated three residential E+ data sets. These data sets contain input and output pairings for E+ simulations, and were built by varying building envelop model parameters. In addition, these data sets each contain 156 building inputs, and two data sets have 90 building outputs. The other residential data set contains only 80 output parameters.

The data set with only 80 output parameters, called Fine Grain (FG), was generated by brute force increments across 14 of the 156 building input parameters. The FG data contains approximately 12,000 simulations, and is approximately 143 gigabytes. Originally the Autotune project was attempting a small brute force pass over the input parameter space, but it was shown to be too computationally expensive. Therefore, the number of brute force building parameters was scaled back to a more computationally reasonable amount.

Given the size of the parameter space, another parameter sweep method was implemented to generate the other data sets. The second data set, called Markov Order 1 (MO1), was generated by running two simulations per input variable. The modified variables were set to a predetermined minimum value and maximum value for these two simulations, while the other variables were set to their average value. These predetermined and average values were selected by building engineers on the Autotune project. This data set is approximately 3.9 gigabytes and contains 299 simulations.

The final data set, called Markov Order 2 (MO2), was generated by running simulations based on all possible minimum and maximum value pairings between two variables. This means we ran four simulations per variable pairing. The simulations covered the (min,min), (min,max), (max,min), and (max,max) pairs. In addition, all

* This project's sole purpose is to build repeatable and transferable automated calibration processes for whole building energy simulators, such as E+. However, the automatic calibration research does not only focus on the E+ simulator.

other variables were set to their average value during the simulation. This data set is approximately 450 gigabytes and contains approximately 28,000 simulations.

2.3 Dealing with Large Datasets

The majority of the presented regression methods have runtimes that are dependent upon the number of training examples. In addition, the memory requirements are dependent upon the number of training examples as well. For example, linear regression requires computing the inverse of the predictors matrix, which is an $O(N^3)$ operation where N is the total number of examples. Additionally, the memory requirements are $O(NM)$ where M is the dimensionality for each observation. This restriction makes scaling standard linear regression to large datasets fairly difficult. Stochastic gradient descent addresses this problem by considering each sample individually, but this method is an approximation and not guaranteed to produce the same solution as the maximum likelihood method, which is unique if the system of equations are well determined.

Similar problems plague the offline versions for all the methods, and are generally solved by online approximation methods. For instance, the general solution for scaling SVR to large datasets is referred to as the working set approach Joachims (1999a). This method sub-samples the data to reduce memory requirements and to reduce the runtime for solving the quadratic optimization problem. Approaches similar to this are required for kernel methods, because the kernel matrix requires $O(N^2)$ memory. In addition, the runtime generally increases as the number of examples increases Shalev-Shwartz and Srebro (2008). However, Shalev-Shwartz and Srebro (2008) notes that for a particular SVM solver and classification problems, the runtime decreases as the number of examples increase. The presented proof only shows this property for linear kernels and for the hinge loss function, a regularization function for classification.

Chapter 3

Sensor-based Modeling

Sensor-based modeling can be viewed as a hybrid between "forward" modeling and "inverse" modeling approaches. This data-driven approach assumes that the sensor data provides a viable model for the entire building – the "forward" component. This means the sensor data encodes the state of weather, building envelope, equipment, and operation schedules over time. Through the application of machine learning algorithms, an approximation of the engineering model is derived statistically – the "inverse" component. However, the machine learning algorithms used by sensor-based modeling allow the data to determine the best model, rather than engineering domain knowledge that may not always be applicable.

Sensor-based modeling serves as an alternative approach to traditional "forward" and "inverse" modeling. In fact, there are numerous sensor-based studies that focus on predicting current and future electrical consumption for commercial buildings Kreider and Haberl (1994); Karatasou et al. (2006); Li et al. (2011). In addition, these studies have established which machine learning techniques perform well at modeling commercial electrical consumption.

While sensor-based modeling is general, the remainder of this chapter focuses on sensor-based energy modeling, which is the application of sensor-based modeling to predicting electrical consumption*.

3.1 Traditional Modeling vs Sensor-Based Modeling

Both forward and inverse modeling approaches, individually, suffer from several problems that are mitigated, if not solved, through sensor-based energy modeling. First, very few design firms have the expertise and can absorb the time and cost necessary to develop a thorough set of inputs during the design phase of a building. Most do so primarily for the largest of projects, despite the fact that the most important energy-consuming decisions are made during this phase and are least costly to remedy during early design. While sensor-based energy modeling does require existing sensor data, and thus implies an existing building, machine learning software trained on data from a similar reference building can function as an approximation engine and may provide sufficiently accurate results for quick feedback during early, iterative building design. Second, there is always a gap between the as-designed and as-built building. During construction, changes are made out of necessity, convenience, or negligence (e.g., lack of insulation in a corner), and many changes are very rarely communicated to designers or energy modelers. Sensors obviously eliminate this problem by measuring the actual state of the building rather than a designer's intentions. Third, sufficient knowledge is rarely available to accurately classify the dynamic specificities of equipment or a given material. Most energy modelers use the *ASHRAE Handbook of Fundamentals* American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc. (2009) to estimate thermal and related properties based on typical values. Many others use the manufacturer's label

* A majority of the work in this chapter was published in a journal paper, Edwards et al. (2012).

information when available. However, few modelers put the materials and equipment through controlled laboratory conditions, or the appropriate ASTM test method, to determine properties of the specimen actually used in the building. The sensor-driven approach can not only capture the current/actual performance of the material, but also its degradation over time. Fourth, traditional modeling approaches can involve manually defining thousands of variables to codify an existing building. Since multiple experts may encode a specific building in many different ways, the large required input space lends itself to problems with reliability/repeatability and ultimately validity. Sensors are much more reliable and repeatable in reporting measured data over time, until a sensor or data acquisition system fails. Fifth, both the inverse statistical model and forward engineering models, by their very nature, necessarily require fixed assumptions and algorithmic approximations. Machine learning allows asymptotic approximation to the "true" model of the data, limited solely by the amount or quality of data provided, the capabilities of the algorithm utilized, or the time available to compute/learn from the available data.

For all its advantages, sensor-based energy modeling also introduces some of its own concerns and limitations. First, the additional cost associated with acquisition and deployment of sensors is not required by previous modeling approaches. Sensor development and costs are dropping with the same transistor density doubling every 18 months as defined by Moore's Law Schaller (1997). Increasingly sophisticated peel-and-stick, wireless mesh, energy-harvesting, system-on-a-chip sensors are becoming readily available. While the increase in capabilities and reduction in costs continue, it is currently not feasible to heavily instrument a building cost-effectively. Second, the number, type, and placement of sensors required to sufficiently capture the state of different building types is an open question. This work addresses this issue through selection of an optimal subset of 140 sensors for predicting hourly energy consumption for three residential buildings, but extrapolation across building types is unproven and sensor counts/types would vary based upon the metric(s) being predicted. It is anticipated that shared, web-enabled databases of heavily instrumented buildings

will help resolve this current issue. Third, sensors, data acquisition systems, and the physical infrastructure upon which they rely can be unstable and result in missing or corrupted sensor data values. To mitigate this real-world issue, intelligent quality assurance and control algorithms Ibargüengoytia et al. (2001) can be applied to detect and/or correct corrupted sensor values. The sensor pre-processing system we currently use notifies assigned personnel via email messages for data channels exhibiting out-of-range errors, using simple statistical tests. Lastly, determining the best machine learning algorithm for a given learning task is an open question. While there exist taxonomies for classifying problem types and appropriate machine learning algorithms Russell and Norvig (2010), rarely is there a known algorithm that is best for solving a given problem (e.g., predicting next hour energy usage). This issue is mitigated by exploring seven different machine learning algorithms and determining which algorithm or algorithms perform best.

We have tested seven different machine learning techniques on our residential data sets, and on the ASHRAE Building Energy Predictor Shootout data set. In this chapter, we briefly outline the technical details for each individual learner. In addition, we discuss advantages, disadvantages, and technical benefits for each technique. We present the techniques in the following order: Linear Regression; FFNN; SVR; Least Squares Support Vector Machines (LS-SVM); HME with Linear Regression Experts; HME with FFNN Experts; and Fuzzy C-Means with FFNN. In addition, the preliminaries chapter commented on the difficulties encountered when scaling these regression methods to large datasets (Section 2.3). The latter discussion is very important, because it highlights a true limitation for most regression methods, and Chapter 5 and 6 show that it is a major problem for construction approximations and relational learning. Note, in the following sections Y refers to the entire set of electrical consumption measurements, y refers to a single consumption measurement, X refers to the entire set of sensor observations, x_i refers to an individual sensor observation, and $\vec{x}$ refers to a vector of sensor observations.

3.2 Problem Statement

Very little sensor-based work focuses on modeling electrical consumption for residential buildings, rather than commercial buildings. In fact, most sensor-based studies conducted with residential buildings model monthly electrical consumption Kolter and Ferreira Jr (2011), while commercial building studies model hourly consumption. This means the few established methods for residential buildings are only tested and verified on monthly data. Therefore, there is a need to explore additional techniques on higher granularity data sets and to establish which machine learning techniques truly perform best at modeling residential electrical consumption.

A problem that hampers sensor-based energy modeling's variability is the associated cost with sensor deployment. Sensor deployment, a non-trivial expense, takes a large amount of planning and installation time, as well as large monetary cost. The human time and equipment cost can be greatly mitigated by reducing the overall number of required sensors within a building. This leaves a key question: which sensors are most important for modeling overall electrical consumption? Removing sensors from the environment can reduce a model's ability to accurately estimate consumption. This means the sensors removed from the environment must be irrelevant or redundant for predicting the electrical consumption. Therefore, it is important to determine which sensors are most important for predicting electrical consumption and reduce cost by eliminating irrelevant or redundant sensors from future buildings.

While solving these two problems shows that sensor-based energy modeling is a viable alternative, forward modeling is still the most general approach. Forward models, such as Energy Plus, can estimate electrical consumption without an existing building or existing sensor information. However, developing an accurate building model takes a significant amount of engineering time. In addition, matching a simulation model with an actual building or a reference building is a slow iterative process, because of the computation time required to check and compare

a model against the expected results. The runtime for simulating an entire year is approximately eight minutes. While the amount of time required to perform 100 iterations over a simulation model (i.e., 800 minutes, not including manual tuning time) may seem reasonable, it could take many more iterations, maybe even thousands. Clearly, there is a need for a method to automate the tuning process and to improve the simulation time required for estimating a year's energy usage.

3.3 Linear Regression

Linear Regression is the simplest technique and can provide a baseline performance measure. Linear Regression is based on fitting a linear function with the following form:

$$y = \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n + \beta_n$$

Here, y is the target value, $x_1, x_2, ..., x_n$ are the available inputs, and β represents the functional weights. While this model is simplistic, it is used to establish a baseline performance for predicting electrical consumption on our residential data sets. If a technique performs worse than the baseline predictor, then it is most likely not appropriate for the data set.

3.4 Feed Forward Neural Network

As mentioned previously, previous studies have shown that Feed Forward Neural Networks (FFNN) are very capable at predicting electrical consumption. These previous studies have leveraged the fact that a FFNN can be used as a general purpose method for approximating nonlinear functions. That is, FFNN can learn to approximate a function f that maps $\Re^m \rightarrow \Re$ without making assumptions about the relationship between the input and outputs.

While a FFNN does not make assumptions about the inputs or outputs, it does require the user to define the model's structure, including the number of hidden layers and hidden units within the network and any other associated parameters. In this work, we explore a FFNN with a single hidden layer, which is the same overall structure as the previous studies. A FFNN with a single hidden layer for function approximation has the following mathematical representation:

$$f(x) = \sum_{j=1}^{N} w_j \Psi_j \left[\sum_{i=1}^{M} w_{ij} x_i + w_{io} \right] + w_{jo}$$

where N represents the total number of hidden units, M represents the total number of inputs, and Ψ represents the activation function for each hidden unit. In this work we selected $\tanh(x)$ as our activation function because prior research has shown good performance using this function Dodier and Henze (2004); Yang et al. (2005); Gonzalez and Zamarreno (2005); Karatasou et al. (2006).

A FFNN's weights are learned using gradient descent-based methods, such as Newton-Raphson, by minimizing a user-specified error function. There are many possible error functions, such as Mean Squared Error (MSE), Sum Squared Error (SSE), and Root Mean Squared Error (RMSE). In this work, we use the SSE function.

However, a gradient descent learning approach poses two problems. The first problem is over-fitting. The FFNN can adjust its weights in such a way that it performs well on the training examples, but it will be unable to produce accurate responses for novel input examples. This problem is addressed by splitting the training set into two parts – a set used for training and a set for validation. When the error increases on the validation set, the learning algorithm should halt, because any further weight updates will only result in over-fitting the training examples.

The second problem involves avoiding local minima and exploring the search space to find a globally optimal solution. A local minimum is a point at which it is impossible to further minimize the objective function by following the gradient, even though the global minimum is not reached. However, it is not possible to

determine if any particular set of weights is a globally optimal solution or a local minimum. It is not possible to completely address this problem, but it is possible to avoid shallow local minima by using momentum and an adaptive learning rate. Momentum incorporates a small portion from the previous weight changes into the current weight updates. This can allow the FFNN to converge faster and to possibly step over shallow local minima. An adaptive learning rate dynamically changes the gradient descent step size, such that the step size is larger when the gradient is steep and smaller when the gradient is flat. This mechanism will allow the learning algorithm to escape local minima if it is shallow enough.

3.5 Support Vector Regression

Support Vector Regression (SVR) was designed and developed to minimize structural risk Smola, A.J. and Schólkopf, B. (2004). That is, the objective is to minimize the probability that the model built from the training examples will make errors on new examples by finding a solution that best generalizes the training examples. The best solution is found by minimizing the following convex criterion function:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l} \xi_i + \xi_i^*$$

with the following constraints:

$$y_i - w^T\varphi(\vec{x_i}) - b \leq \epsilon + \xi_i$$

$$w^T\varphi(\vec{x_i}) + b - y_i \leq \epsilon + \xi_i^*$$

In the above equations, ϵ defines the desired error range for all points. The variables ξ_i and ξ_i^* are slack variables that guarantee that a solution exists for all ϵ. C is a penalty term used to balance between data fitting and smoothness. Lastly, w are the

weights for the regression, and φ represents a kernel function for mapping the input space to a higher dimensional feature space.

There is one major advantage within the SVR optimization formulation; there is a unique solution which minimizes a convex function. However, the unique solution is dependent upon providing C, ϵ, and the necessary parameters for the user-selected kernel function φ. There are many techniques for selecting the appropriate parameters, such as grid search with cross-validation, leave-one-out cross-validation, and many more. The work of Smola, A.J. and Schólkopf, B. (2004) provides a detailed overview of the different tuning techniques. In this work, all parameter settings were determined via grid search with cross-validation using LIBSVM's provided utilities Chang and Lin (2011a).

However, SVR does have a potential disadvantage: scalability. The convex criterion function is optimized using quadratic programming optimization algorithms. There are many different algorithms and each has its own advantages and disadvantages Smola, A.J. and Schólkopf, B. (2004), but the primary disadvantages are generally memory requirements and speed. However, the data sets used in our work are not large enough for these issues to be a real concern.

3.6 Least Squares Support Vector Machine

Least Squares Support Vector Machine (LS-SVM) is very similar to SVR, but with two notable differences. The first difference is the criterion function, which is based on least squares. The second difference is that the problem constraints are changed from inequality to equality. These differences allow the optimization function to be formulated as:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i^2$$

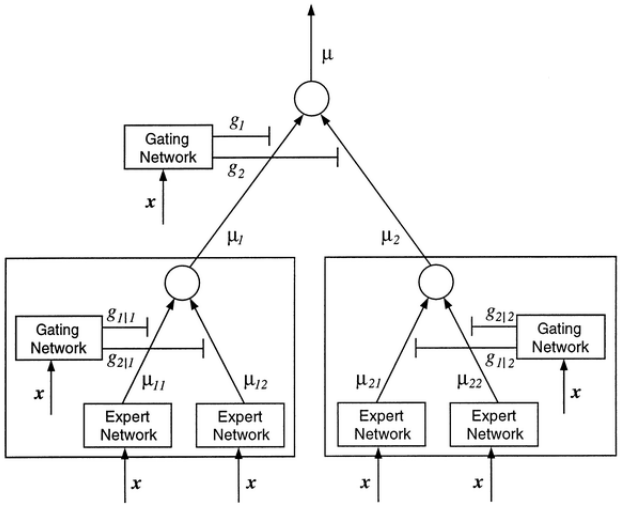


Figure 3.1: An example Hierarchical Mixture of Experts model with depth 2 and branching factor 2. This figure is provided by Jordan and Jacobs (1994).

with the following constraint:

$$w^T \varphi(\vec{x_i}) + b + \xi_i = y_i$$

One advantage LS-SVM has over SVR is that this modified criterion function does not require quadratic programming to solve the optimization problem. This allows LS-SVM to find solutions much faster by solving a set of linear equations. The set of linear equations and their solution are well documented in Suykens et al. (2002a). However, LS-SVM uses all data points to define its solution, while SVR only uses a subset of the training examples to define its solution. This means that LS-SVM loses the sparsity property, which may or may not affect the solutions' ability to generalize. However, there are studies that address the sparsity issue through pruning or via weighting the examples Suykens et al. (2002b); Hoegaerts et al. (2004).

3.7 Hierarchical Mixture of Experts

Hierarchical Mixture of Experts (HME) is a type of Neural Network that learns to partition an input space across a set of experts, where the input space in our application is the raw sensor values. These experts will either specialize over a particular region or assist each other in learning a region or regions. These HME models are very useful for exploring the possibility that a data set contains multiple regimes or sub-populations. For example, a residential home's electrical consumption can vary according to the seasons – fall, winter, spring, and summer. These variations may be best explained by separate individual models. An HME model tries to discover these different sub-models automatically, and fit an Expert to each sub-model. While the previous motivating example implies temporal based sub-model changes, the HME model can only detect sub-model changes by using spatial differences, as well as using each expert's ability to produce accurate predictions during training.

HME models are constructed using two types of networks: Gating and Expert networks. Figure 3.1 presents an example HME with two layers of Gating networks and four Expert networks. This particular HME is modeled as:

$$\mu = \sum_{i} g_i \sum_{j|i} g_{j|i} F_{ji}(\vec{x})$$

where g_i represents the top level gating network's output, $g_{j|i}$ represents the outputs from the lower level gating networks, and $F_{ji}(\vec{x})$ represents the output from an Expert network. This example model is easily extended to have additional Gating networks and Experts by adding additional summations.

The Gating network probabilistically partitions the input space across either additional Gating or Expert networks. The partitioning is achieved using the following softmax function:

$$g_i = \frac{e^{\xi_i}}{\sum_{k=1}^{N} e^{\xi_k}}$$

where ξ represents the Gating network outputs, g_i is the normalized weight associated with the ith sub-network, and N represents the total number of sub-networks. Each Gating network approximates the conditional probability $P(Z|X)$ in which Z represents the set of direct sub-networks and X represents the set of observations. Approximating $P(Z|X)$ allows the Gating network to determine which Expert network or networks is more likely to produce an accurate prediction.

Each Expert network represents a complete learning system. However, unlike a standalone learning system, each Expert is expected to specialize over different regions defined by the Gating networks. In the original HME studies, the only supported expert learner was Neural Networks Jordan and Jacobs (1992). However, a later extension on the work introduced support for Linear Regression Experts Jordan and Jacobs (1994). While these studies only presented Neural Network and Linear Regression Experts, the learning procedures introduced in the extension do not limit the Experts to only these learning systems. The only restriction placed on the Experts is that they have an associated likelihood function. For example, the assumed likelihood function in these previous studies for regression problems is that each Expert's error rate follows a Gaussian distribution.

The original studies present three different maximum likelihood learning algorithms. The first algorithm is based on using gradient ascent. Using the HME shown in Figure 3.1 as an example, all three algorithms attempt to maximize the following likelihood function:

$$L(Y|X,\theta) = \prod_t \sum_i g_i^{(t)} \sum_j g_{j|i}^{(t)} P_{ij}(y^{(t)}|\vec{x}^{(t)}, \theta_{ij})$$

where P_{ij} represents an Expert's likelihood function and θ represents parameters associated with each Gating network and with each Expert.

The other two algorithms approach the problem as a maximum likelihood problem with missing data. The missing or unobservable data is a set of indicator variables that specify the direction for partitioning the input space at each Gating network.

If all indicator variables are known, then maximizing the HME's likelihood function is split into two separate problems Jordan and Jacobs (1994). The first problem is learning the parameters for each individual Gating network, while the second problem is training each Expert on the appropriate training examples. Given that it is generally impossible to know the exact value for each indicator variable in advance, the original developers derived two different Expectation Maximization (EM) Dempster et al. (1977) algorithms. The first algorithm is an exact EM algorithm and the second algorithm approximates the first algorithm.

In addition to FFNN and Linear Regression Experts, we extended the Mixture of Experts (MoE) with LS-SVM Experts, by Lima et al. (2009), to Hierarchical Mixtures. The Maximization process is presented as a weighted regression problem in both HME EM algorithms, which implies any learning system that supports weighted examples can be used as an Expert. We utilize this property and the robust LS-SVM work by Suykens et al. (2002b) to integrate LS-SVM Experts into the HME framework. However, we found that the results for HME with LS-SVM on our residential data set and the Great Energy Prediction Shootout data set were not statistically different from a single LS-SVM. We believe this is due to all LS-SVM Experts using the same parameter settings as the single LS-SVM model. The findings in Lima et al. (2009) suggest that the parameter settings can be the same across the LS-SVM Experts, but the parameter settings should be determined by searching the parameter space using the entire mixture model.

3.8 Fuzzy C-Means with Feed Forward Neural Networks

An alternative approach to HME is to separate the learning process into two steps. The first step is an unsupervised learning phase that uses clustering to approximate $P(Z|X)$, and the second step is to use each cluster to train the Experts. It is

possible to use any clustering algorithm, such as K-Means, Self-Organizing Maps, Hierarchical Clustering, etc. However, a clustering algorithm that does not allow observations to belong to multiple clusters will produce very rigid approximations. A rigid approximation will cause Experts to ignore large sets of observations, which can cause the Experts to produce very poor models. This means each Expert will be less likely to produce reasonable responses when accounting for errors in the approximated $P(Z|X)$. We avoid rigid approximations by using Fuzzy C-Means (FCM), which allows for observations to belong to multiple clusters.

FCM is based on minimizing the following criterion function:

$$\sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^{m}\|\vec{x}_i - \vec{c}_j\|^2$$

where u_{ij} represents the probability that $\vec{x}_i$ is a member of cluster $\vec{c}_j$, and m is a user-defined parameter that controls how much an observation can belong to multiple clusters. The criterion function is minimized through an iterative process using the following equations:

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^{m}\vec{x}_i}{\sum_{i=1}^{N} u_{ij}^{m}}$$

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \frac{\|\vec{x_i}-\vec{c_j}\|}{\|\vec{x_i}-\vec{c_k}\|}^{\frac{2}{m-1}}}$$

Iterating over the above equations will produce N cluster centroids and a weight matrix U. N represents the total number of user-defined clusters and each row in U represents an instance of $P(Z|X)$. The weight matrix can be used to train a Gating network or for weighting the training examples when fitting the Experts. In this work, we choose to use the second option, and use N cluster centers to approximate $P(Z|X)$ for new observations by computing the second equation.

While we implemented FFNN, Linear Regression, and LS-SVM Experts for the HME models, we have only explored FFNN Experts for this two-step approach. This approach is not limited to FFNN Experts, and can support all learning

systems that can incorporate weighted training examples. In addition, the likelihood function requirement for the Experts is removed. While this approach seems superior to the HME, it relies on the critical assumption that the spatial relation between observations can approximate $P(Z|X)$, while HME approximates $P(Z|X)$ by maximizing $P(Y|X,\theta)$.

3.9 Temporal Dependencies

In the realm of function approximation, *temporal dependencies* means that the target response y_t is dependent on past observations, x_{t-1}, as well as current observations x_t. These temporal dependencies either follow a Markov order or are sparse. If the dependencies follow a Markov order, then the response y_t is dependent on previous complete sets of observations. For example, if y_t has temporal dependencies with Markov Order 2, then it is dependent on x_t, x_{t-1}, x_{t-2}. However, sparse temporal dependencies indicate that y_t can be dependent on any combination of past observations rather than a complete set. Exploring all possible sparse temporal dependencies grows exponentially and is thus intractable.

Our work focuses on predicting future hourly electrical consumption. This means we can only use observations x_{t-1}, x_{t-2}, etc., to predict y_t. If we did not follow this constraint, we would use future information to predict y_t. Therefore, Markov order 1 models use observation x_{t-1}, order 2 models use observations x_{t-1} and x_{t-2}, and so forth.

In previous works, researchers explored sparse temporal dependencies either with manual statistical testing or automatically, by defining a feasible search space within the learning system. The winner for the first Shootout, which we discussed previously, used automatic relevance detection (ARD; Section 4.4) to automatically determine the relevant inputs. The possible inputs included temporal dependencies. However, the total number of available inputs for the competition was fairly small. For example, the winner's FFNN used 25 different inputs, while a single order 3 model uses

approximately 432 inputs. Therefore, we only consider the entire set of inputs, rather than trying to search for the best inputs. However, the feature selection methods in Chapter 4 are able to facilitate sparse temporal dependency detection.

3.10 Model Selection

Each previously mentioned learning system has a variety of different parameters. Some parameters are estimated during the learning process, while others are user-defined parameters. In addition, each different combination of learned parameters and user-defined parameters constitutes a single model configuration. In order to determine which learning system performs best at predicting residential electrical consumption, we need to select the best model configurations for each technique and compare these best configurations. This type of comparison facilitates a fair comparison across all techniques.

There are several different model selection techniques. For example, cross-validation methods help find parameter estimates that can generalize to unseen data by periodically testing the current model on a validation set. Another cross-validation method, called K-Folds cross-validation, ensures that each data point is used as a testing example at least once, and that the training and testing sets are fixed. This means that each learning system can be compared using the same testing and validation sets, which is ideal for determining how different user-defined parameters affect the models.

We use a combination of cross-validation and K-Folds cross-validation to select the best predictive model for each technique. We separate out a cross-validation set from the allocated training data, which leaves each learning system with a training set, a validation set, and a testing set. However, the Linear Regression models do not utilize the validation set, because the parameters are estimated using a non-iterative maximum likelihood method. We then select the model from each technique that has the best performance across all the testing sets. This allows us to identify models that

generalize well to unseen data, and determine which user-defined parameters settings are best for each learning system.

3.11 Performance Metrics

The primary measure for selecting the winners in the ASHRAE competition was the Coefficient of Variance (CV) measure Kreider and Haberl (1994), which determines how much the overall prediction error varies with respect to the target's mean. A high CV score indicates that a model has a high error range. The CV measure is defined as follows:

$$\text{CV} = \frac{\frac{1}{N-1}\sqrt{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}}{\bar{y}} \times 100$$

where $\hat{y}_i$ is the predicted energy consumption, y_i is the actual energy consumption, and $\bar{y}$ is the average energy consumption.

A second metric, Mean Bias Error (MBE), was used to break ties within the competition. This metric establishes how likely a particular model is to over-estimate or under-estimate the actual energy consumption. A MBE closest to zero is preferred, because this means the model does not favor a particular trend in its prediction. The MBE measure is defined as follows:

$$\text{MBE} = \frac{\frac{1}{N-1}\sum_{i=1}^{N}(y_i - \hat{y}_i)}{\bar{y}} \times 100$$

where $\hat{y}_i$, y_i, and $\bar{y}$ represent the same components presented in the CV measure.

Another metric that is commonly used in the literature to assess regression accuracy is Mean Absolute Percentage of Error (MAPE) Karatasou et al. (2006); Gonzalez and Zamarreno (2005). The MAPE measure determines the percentage of error per prediction, and is defined as follows:

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y}_i|}{y_i} \times 100$$

Table 3.1: Great Energy Prediction Shootout Results. Best results are shown in bold font.

S1

	CV(%)	MBE(%)	MAPE(%)
Regression	14.12±0.00	7.69±0.00	13.41±0.00
FFNN	**11.29±0.00**	**8.32±0.00**	**9.14±0.00**
SVR	11.93±0.00	8.95±0.00	9.63±0.00
LS-SVM	13.70±0.00	10.32±0.00	11.21±0.00
HME-REG	14.11±0.00	7.66±0.00	13.40±0.00
HME-FFNN	11.49±0.00	2.91±0.00	9.73±0.00
FCM-FFNN	11.51±0.00	8.71±0.00	9.45±0.00

S2

	CV(%)	MBE(%)	MAPE(%)
Regression	4.07±0.00	1.01±0.00	2.86±0.00
FFNN	2.93±0.00	0.64±0.00	1.77±0.00
SVR	3.97±0.00	1.41±0.00	2.31±0.00
LS-SVM	6.35±0.00	1.53±0.00	4.50±0.00
HME-REG	4.05±0.00	0.99±0.00	2.85±0.00
HME-FFNN	**2.75±0.00**	**0.52±0.00**	**1.60±0.00**
FCM-FFNN	**2.71±0.00**	**0.55±0.00**	**1.61±0.00**

where $\hat{y}_i$ and y_i represent the same components defined in the CV and MBE measures.

In this work, we use CV as our primary metric. MBE is the first tie breaker, and MAPE is the final tie breaker. We only take the tie breaker metrics into consideration when the CV metric does not measure a statistical difference between two techniques. If both original ASHRAE metrics are inconclusive, our decisions are based on the MAPE metric.

3.12 Predicton Results

Our experimental results are organized in the following order: ASHRAE Shootout 1, Campbell Creek House 1, Campbell Creek House 2, and Campbell Creek House 3. Each subsection presents the best performing models from the seven techniques. Following these result sections, we present a results summary, which presents the best general overall technique and highlights the key results for each data set.

3.12.1 Great Energy Prediction Shootout

For comparison purposes, we ran our seven implemented machine learning techniques on the earlier Great Energy Prediction Shootout data set. The results for these experiments are presented in Table 3.1. We are not able to make statistical claims about the difference between techniques, because the original competition presented

Table 3.2: Results for all techniques applied to Campbell Creek House 1. Best results are shown in bold font.

House 1

Order 1

	CV(%)	MBE(%)	MAPE(%)
Regression	32.38±1.91	-0.06±1.08	30.52±1.41
FFNN	25.10±2.34	0.66±1.43	21.08±1.14
SVR	24.60±1.78	-2.46±0.95	17.05±0.94
LS-SVM	23.39±1.26	0.01±0.84	18.21±0.89
HME-REG	32.35±1.82	-0.05±1.02	30.57±1.42
HME-FFNN	22.77±1.56	0.15±0.98	17.74±0.65
FCM-FFNN	22.65±1.42	0.81±0.95	18.18±0.75

Order 2

	CV(%)	MBE(%)	MAPE(%)
Regression	27.63±1.95	-0.03±1.09	26.18±1.51
FFNN	24.32±2.61	0.53±1.74	22.28±2.67
SVR	21.58±1.40	-1.41±0.89	16.41±0.95
LS-SVM	**20.05±0.81**	**0.06±0.62**	**16.11±0.85**
HME-REG	27.60±2.13	-0.03±1.01	26.11±1.67
HME-FFNN	20.15±1.65	0.46±0.93	17.07±1.19
FCM-FFNN	20.53±1.76	0.74±0.87	17.57±1.42

Order 3

	CV(%)	MBE(%)	MAPE(%)
Regression	26.27±1.19	-0.11±1.45	24.33±0.96
FFNN	25.24±1.59	1.00±1.05	22.29±1.81
SVR	21.32±1.32	-1.50±0.80	15.48±0.87
LS-SVM	20.36±1.46	0.11±0.63	15.73±1.11
HME-REG	26.14±1.10	-0.08±1.44	24.21±0.93
HME-FFNN	20.39±1.67	0.70±0.92	17.09±0.81
FCM-FFNN	21.03±1.29	0.47±1.49	18.27±1.06

only a single training and testing set. However, the S1 results indicate that a FFNN is the best predictor for electrical consumption. This finding is consistent with the existing literature Kreider and Haberl (1994). However, all methods except Linear Regression, HME with Linear Regression, and LS-SVM are competitive with the best three competition winners: CV – 10.36%, 11.78%, 12.79%.

The S2 results in Table 3.1 suggest that HME with FFNN and FCM with FFNN are better than the FFNN. The existing published results for the S2 inputs range from 2.44% to 3.65% Karatasou et al. (2006); Li et al. (2011). From these results, we can conclude that Neural Network type methods perform best on this data set. We can also conclude that LS-SVM is the worst advanced technique, with Linear Regression and HME with Linear Regression being only slightly better.

3.12.2 Campbell Creek House 1

Table 3.2 presents the results from applying all the techniques to House 1 with different Markov orders. These results illustrate which techniques perform the best

on House 1 and the effects that different Markov orders have on these techniques. Almost all techniques increase in performance as the order increases. The three methods that do not increase in performance are FFNN, HME with FFNNs, and FCM with FFNNs. The FFNN results are not statistically different across all orders. The other two techniques show performance increases with order 2, but order 3 is not statistically different.

According to the CV metric, the best techniques are the order 2 SVR, order 2 LS-SVM, order 2 HME with FFNNs, and order 2 FCM with FFNNs. While the CV performance for the SVR model is not significantly different, its MBE error is statistically different from the other techniques, illustrating that it has potential to perform much poorer than the other three techniques. In addition, the other three techniques do not have significantly different MBE results. Even though the second tie-breaker metric does not indicate a single best model, the third tie-breaker (MAPE) shows clearly that LS-SVM has the best MAPE measure and is statistically different from HME with FFNNs and FCM with FFNNs. Therefore, LS-SVM is the best model for predicting next hour energy consumption for House 1.

3.12.3 Campbell Creek House 2

The results for House 2 (Table 3.3) show a different performance trend as the Markov order increases, compared to House 1. While most techniques illustrated an increase in performance on House 1 as the order increased, these techniques only present small improvements on House 2. The improvements are only statistically significant for the baseline Linear Regression technique and order 3 SVR.

Given the minimal performance gains from the increasing orders and the CV results for House 2, the best techniques are order 1 LS-SVM and Order 1 FCM with FFNNs. The order 1 models are selected over the Order 2 and 3 models, because the three models are not statistically different within an acceptable confidence, and higher order models are much more complex. The higher order models are more complex

Table 3.3: Results for all techniques applied to Campbell Creek House 2. Best results are show in bold font.

House 2

Order 1

	CV(%)	MBE(%)	MAPE(%)
Regression	36.73±2.26	-0.13±1.00	31.01±3.48
FFNN	33.24±1.26	0.50±0.91	27.28±3.12
SVR	30.36±1.83	-2.95±1.03	20.44±2.81
LS-SVM	**27.88±1.24**	**-0.05±0.91**	**20.47±2.37**
HME-REG	35.82±1.04	0.15±0.88	30.48±3.20
HME-FFNN	29.30±1.28	0.09±1.01	22.71±2.92
FCM-FFNN	**28.14±1.21**	**0.40±0.97**	**21.96±2.74**

Order 2

	CV(%)	MBE(%)	MAPE(%)
Regression	34.15±1.66	0.05±1.61	28.36±3.72
FFNN	33.83±1.98	0.21±1.45	27.07±4.14
SVR	29.22±1.06	-3.00±1.12	19.42±3.27
LS-SVM	27.43±1.90	0.20±1.03	20.17±2.26
HME-REG	34.15±1.74	0.14±1.38	28.29±3.86
HME-FFNN	28.17±2.04	0.26±0.58	22.43±2.44
FCM-FFNN	28.34±1.67	-0.20±1.27	22.30±3.28

Order 3

	CV(%)	MBE(%)	MAPE(%)
Regression	33.15±1.33	-0.02±0.96	27.87±2.40
FFNN	34.23±1.63	2.01±2.45	29.62±2.16
SVR	28.59±2.05	-2.33±1.09	19.58±2.07
LS-SVM	27.68±1.91	-0.02±1.71	20.23±2.56
HME-REG	33.20±1.32	-0.08±0.97	27.95±2.31
HME-FFNN	29.64±2.21	-0.12±1.64	24.81±0.38
FCM-FFNN	28.94±1.46	0.45±1.27	22.76±2.03

because as the number of inputs increases, the total number of parameters to estimate increases. A more complex model has less potential to generalize to new examples, which makes it less desirable when simpler models provide equal performance. In addition, the tie breaker measures MBE and MAPE are not statistically different for all orders.

3.12.4 Campbell Creek House 3

The results for House 3, shown in Table 3.4, present the same trend as the House 2 results. As the order increases, most techniques have minimal or no performance gains. The only models that present statistically significant improvements are order 3 SVR and order 2 LS-SVM. The order 3 SVR shows improvement in the CV measure, while the order 2 LS-SVM presents improvement in the MAPE measure. All other models are not statistically different within a reasonable confidence range across the different orders.

According to the results in Table 3.4, order 3 SVR's CV value is statistically different from every model except order 2 and 3 LS-SVMs' CV values. In addition,

Table 3.4: Results for all techniques applied to Campbell Creek House 3. Best results are shown in bold font.

House 3

Order 1

	CV(%)	MBE(%)	MAPE(%)
Regression	40.07±2.21	0.07±1.15	32.49±1.88
FFNN	37.15±1.57	0.35±2.03	28.92±2.55
SVR	33.71±1.72	-3.36±0.99	21.49±1.80
LS-SVM	31.60±2.07	-0.15±1.10	22.25±1.33
HME-REG	39.17±2.17	0.33±1.38	31.72±2.07
HME-FFNN	32.98±1.28	-0.12±0.84	23.99±1.63
FCM-FFNN	33.03±1.67	0.93±1.52	25.28±2.14

Order 2

	CV(%)	MBE(%)	MAPE(%)
Regression	39.26±4.19	0.11±1.86	31.34±2.58
FFNN	38.02±2.49	2.05±2.67	29.83±2.02
SVR	32.38±2.96	-3.12±1.73	20.72±1.38
LS-SVM	**30.66±2.53**	**-0.05±0.93**	**21.33±1.40**
HME-REG	38.48±4.34	1.03±1.72	30.53±3.07
HME-FFNN	32.99±2.17	1.07±1.17	24.76±1.94
FCM-FFNN	32.92±2.49	0.76±2.03	24.20±2.06

Order 3

	CV(%)	MBE(%)	MAPE(%)
Regression	38.53±3.47	0.15±1.22	30.49±2.15
FFNN	38.58±2.07	-0.08±2.46	30.57±2.51
SVR	31.88±2.01	-2.84±0.97	20.47±1.69
LS-SVM	30.78±2.56	-0.21±1.04	21.36±1.50
HME-REG	38.22±3.58	1.20±1.49	29.52±2.47
HME-FFNN	33.34±1.83	1.09±1.24	25.15±2.13
FCM-FFNN	33.66±2.09	1.17±1.30	25.51±1.72

order 1 LS-SVM's CV value is not statistically different from all HME with FFNN models and FCM with FFNN models, but the CV values for orders 2 and 3 are statistically better. Therefore, order 2 LS-SVM and order 3 SVR are the best models based on the CV measure. The order 3 LS-SVM model is excluded because it is not statistically different from the simpler order 2 model.

Note that the House 3 results indicate that SVR demonstrates a large MBE measure for all Markov orders. This means that the SVR model is removed from consideration based on the second tie-breaker measure. Therefore, the best technique for predicting next hour energy consumption for House 3 is LS-SVMs.

3.13 Results Summary

Our findings indicate that FFNN performs best on the original ASHRAE Shootout data set, which is consistent with the literature. However, our results for S2 indicate that other Neural Network methods might perform better. This is consistent with the recent work in Li et al. (2011).

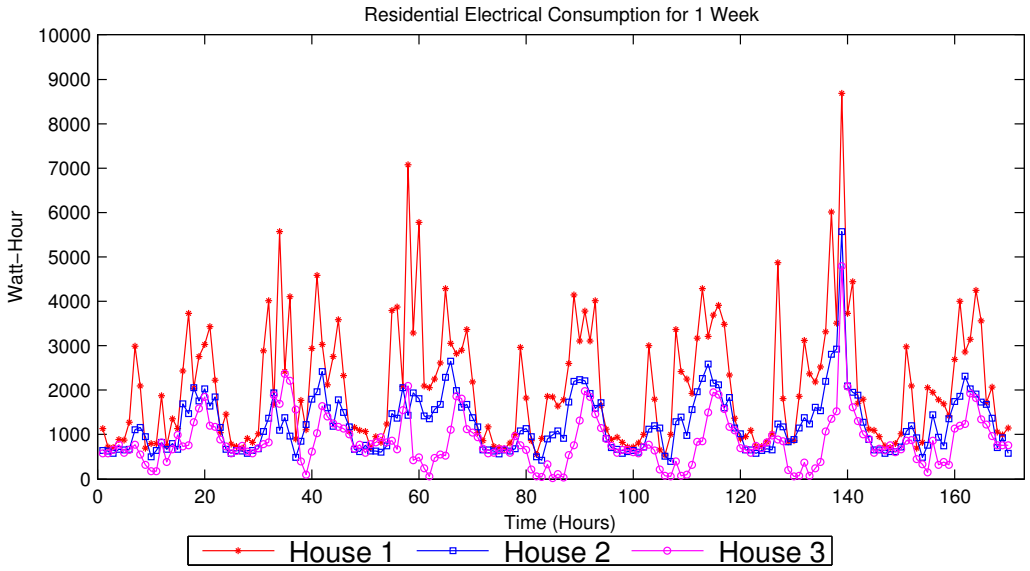


Figure 3.2: This figure presents one week of electrical consumption for all three residential homes, from the second week in September, 2010.

Our findings also indicate that traditional methods, such as FFNN, are not the best overall method for predicting future residential electrical consumption. In fact, on House 3 the FFNN's performance is extremely close to the baseline performance established by Linear Regression. Traditional methods perform better on House 1 and 2, but not as well as other techniques.

Despite traditional methods not performing as well on the residential data sets, our results establish that FCM with FFNN, HME with FFNN, and LS-SVM work well on all three houses. However, LS-SVM is statistically the best technique at predicting future residential electrical consumption over the next hour.

3.14 Discussion

The different performance results for each house stem from the fact that each house is fundamentally different. These physical differences make each house have a very

different energy response pattern, even though each house is automated to run exactly the same schedule. Figure 3.2 illustrates the electrical consumption for a single week in September. The complexity of the energy patterns exhibited by Houses 2 and 3 make them harder to predict than House 1. The figure shows that House 3 is prone to sudden drops in electrical consumption, while House 2's electrical consumption fluctuates much more frequently. House 1 may appear to fluctuate as sharply as House 2, but the fluctuations are much less on average. The physical differences certainly impact the physical sensor data as well.

The results from the Great Energy Predictor Shootout and results from predicting electrical consumption in other commercial buildings have established expected ranges for good CV values – on the order of 2% to 13%, according to the existing literature. The results are clearly dependent on the input variables, but a learning approach is generally considered acceptable if it is within that range. However, we note that our residential results are not within this range. These results are not due to the learning approaches being implemented incorrectly or poorly. In fact, all learning approaches are implemented using existing or modified software packages. The LS-SVM implementation is from LS-SVMlab Suykens et al. (2002a), the SVR implementation is from LIBSVM Chang and Lin (2011a), the HME implementation uses modified software provided by the authors of Martin et al. (2004), and all remaining learning systems are implemented using existing MATLAB modules provided by Mathworks. Considering the reasonable performance of these same techniques on the Great Energy Prediction Shootout data set (Table 3.1) and the fact that all techniques are built using established software, the only possible cause for not matching the established CV range is that each house has more complex energy usage patterns than typical commercial buildings.

Comparing the residential electrical consumption (Figure 3.2) with the commercial electrical consumption (Figure 3.3) shows that commercial buildings have fairly stable usage patterns and less sudden change than residential buildings. The reason for this difference is based purely on the size of the buildings and the fact that small variations

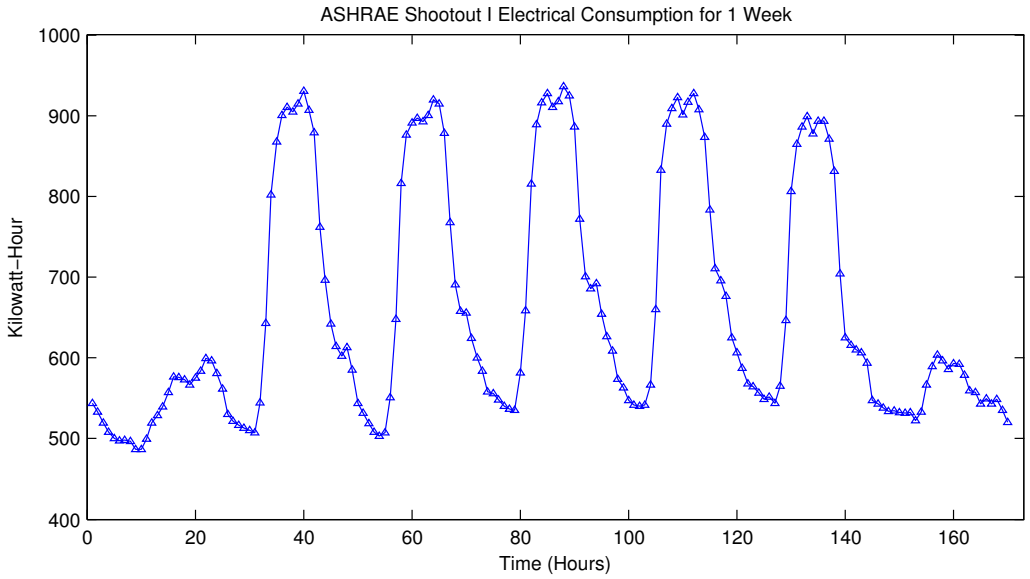


Figure 3.3: One week of electrical consumption for the Great Energy Prediction Shootout building, from the second week in September, 1989.

in consumption do not significantly affect the overall consumption. A larger building will obviously consume more electricity and contain more people, which means that the actions of a few individuals turning on lights or using additional electricity will have very little effect on the buildings' consumption trend. However, in a smaller building, minor changes to the environment can cause noticeable effects. For example, turning all the lights on in most houses will cause more noticeable fluctuation than turning on the equivalent number of lights in a commercial building.

In addition, residential buildings exhibit more complex usage patterns. Figure 3.4 illustrates three weeks of measured electrical consumption for House 3. The usage patterns are very similar for the first two weeks and share similar highs and minimums. However, the usage pattern completely changes during the third week (hours 315 through 500). This variability is mostly dependent upon the house's ability to produce solar power and how much solar power the house is able to produce. While this figure illustrates changes in consumption patterns for House 3, changes in

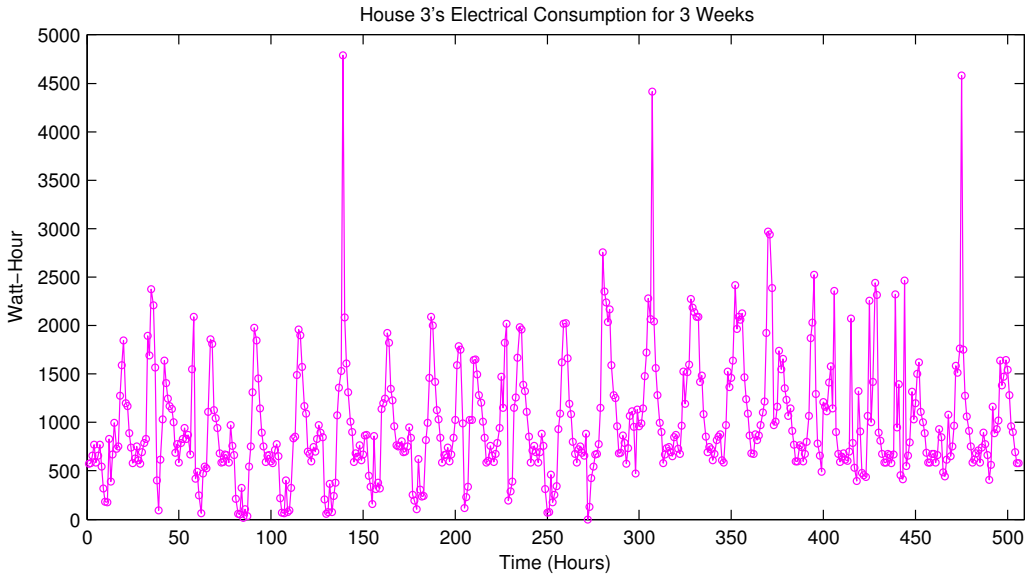


Figure 3.4: Three weeks of electrical consumption for House 3, starting from the second week in September, 2010.

consumption patterns are not unique to House 3 and also occur in Houses 1 and 2; the pattern changes are just more pronounced in House 3.

The Great Energy Prediction Shootout data set does contain changes in consumption patterns, but these changes correspond with holidays, weekends, and normal vacation periods. On the other hand, the changes in these residential homes is dependent on environmental variables and changes in occupant behavior. Thus, these three homes provide a rich and interesting data set for modeling energy prediction that is more challenging than the currently available commercial data sets.

According to the results presented in Tables 3.2, 3.3, and 3.4, changing the Markov order had varying affects. Most techniques applied to House 1 showed a statistically significant performance increase as the order was increased from 1 to 2. On House 1, fewer techniques present improvement by increasing the order even further. However, most techniques applied to Houses 2 and 3 show very little or no performance gains as the order increases. On House 2 only Linear Regression shows

statistically significant improvements by increasing the order. In addition, only two techniques show statistically significant improvement on House 3: LS-SVM and SVR.

There are two possible explanations for these results. First, the temporal dependencies could extend back much further in time than order 3. Second, the consumption patterns could change often enough that increasing the past observations does not help predict future consumption. The first option is possible, but requires further testing and evaluation. However, extending the order further without removing irrelevant inputs may cause most models to perform worse than the ones with smaller orders, due to overfitting. Therefore, this requires testing higher orders and determining the most relevant inputs for predicting electrical consumption. The feature selection methods explored in this dissertation enable relevant input selection, but those methods were not used to generate these results.

The second option is the most plausible explanation. Houses 2 and 3 change consumption patterns fairly often and are dependent on future events that are not always represented within past observations. For example, House 3's ability to generate solar power is dependent on external weather events that are not guaranteed to follow a regular pattern. However, House 2 is more difficult to explain. House 2's consumption pattern changes regularly, except that there are periods where the electrical consumption sporadically increases more than the normal trends. These instantaneous changes in patterns are not represented by past observations, which means increasing the order will not necessarily help.

Our residential results establish that LS-SVM is the best technique from the ones we explored. However, the Shootout results establish that this technique only performs better than HME with Linear Regression and Linear Regression alone. Clearly the LS-SVM model fails to generalize to the Shootout testing data. The model failed to generalize because the provided training data is not general. The electrical response signal for the training data and testing data are statistically different, but LS-SVM uses every training example to help define its model. This means that the LS-SVM builds a model that expects the testing response to resemble

Table 3.5: Great Energy Prediction Shootout results using 3-Folds. The data set's order was randomized before being divided into folds. Each test set has approximately the same number of examples as the original competition test set. Best results are shown in bold font.

ASHRAE Shootout

S1

	CV(%)	MBE(%)	MAPE(%)
Regression	13.26±0.16	-0.02±0.43	11.64±0.11
FFNN	**8.81±0.17**	**0.01±0.10**	**7.10±0.09**
SVR	9.16±0.23	0.05±0.04	7.48±0.12
LS-SVM	**8.85±0.18**	**0.02±0.21**	**6.95±0.21**
HME-REG	13.26±0.15	0.03±0.41	11.65±0.10
HME-FFNN	**8.74±0.22**	**-0.02±0.04**	**7.00±0.11**
FCM-FFNN	**8.74±0.26**	**0.05±0.24**	**6.99±0.21**

S2

	CV(%)	MBE(%)	MAPE(%)
Regression	4.01±0.35	0.00±0.27	2.71±0.08
FFNN	2.29±0.16	0.06±0.12	1.51±0.05
SVR	3.27±0.36	0.09±0.16	1.90±0.12
LS-SVM	3.77±0.44	-0.07±0.08	2.13±0.20
HME-REG	4.01±0.35	0.01±0.29	2.70±0.10
HME-FFNN	**2.20±0.19**	**-0.03±0.07**	**1.39±0.01**
FCM-FFNN	**2.17±0.17**	**0.01±0.11**	**1.38±0.00**

the observed training reponse. However, in this situation the electrical consumption pattern changes and the LS-SVM model is not able to predict these changes. We were able to test this idea by randomizing the Shootout training and testing data, such that the sets were more similar.

Our experiments with this modified data set show a performance increase for most techniques (Table 3.5). More importantly, LS-SVM is now a more competitive learning algorithm on this data set when presented with a more general training set. In our residential experiments, we shuffled the data sets before dividing the data into folds. This allowed us to perform all experiments with training and testing data sets that covered a wide range of different scenarios. Ultimately, we plan to train all methods on the entire 2010 Campbell Creek data set and perform tests on the entire 2011 Campbell Creek data set once the year is complete.

Chapter 4

Sensor Selection

Feature Selection is viewed as a Model Selection problem within the machine learning and statistical community, and involves trying to select a set of inputs, variables or functions that produces an optimal predictive model. An optimal model will generalize well to new examples and has little bias towards the training examples. There are many different Model Selection techniques for approximating which model has the least amount of bias; each technique has advantages and disadvantages. However, this chapter only focuses on a few types of Model Selection techniques that assume parsimony or Occam's razor governs the best model. These types of Model Selection, or Feature Selection, techniques can be categorized into three types – Filter, Wrapper, and Embedded Guyon and Elisseeff (2003).

Filter algorithms provide a preprocessing approach to reduce model complexity. A filter algorithm attempts to discover relevant inputs based on the statistical information presented within the data. For example, a common filter approach is to apply Principle Component Analysis (PCA) to a dataset and use the resulting coefficients in the transformation matrix to derive a variable ranking. Given this ranking, one can pick a fixed number of variables to use for the learning problem. The main advantage provided by Filter methods is that they are relatively computationally inexpensive, making them much faster to use than the other method types. However,

these approaches will only provide heuristic estimates for selecting the correct variables to use in the model and cannot provide any provable properties.

Wrapper methods perform model selection by attempting to reduce the number of parameters used by the learner external to the learning system. Essentially, Wrapper techniques provide a method for searching through different parameter configurations and use the learning system to judge the quality of these configurations. These types of methods generally perform much better than Filter methods at selecting the truly important variables, because the variables are selected with respect to the task; as mentioned previously, Filter methods only use the statistical information within the data to guide variable selection. Although Wrapper methods are more computationally expensive, the benefits provided by a guided search are generally worth the computational cost.

The last category, Embedded algorithms, perform model selection by indirectly selecting the appropriate variables required to perform the specified task. This approach ignores irrelevant inputs without explicitly searching for the relevant inputs. For example, a learning algorithm that uses an embedded method can drive the weights given for relevant variables towards large values, and the weights for irrelevant variables towards zero. This class of learning algorithms can have mixed reliability, and ultimately provide heuristic guidance for selecting the relevant variables. However, this heuristic guidance is more directed than the Filter methods, and the methods can be computationally more efficient than the Wrapper methods.

Given the advantages and disadvantages for each Feature Selection algorithm category, this chapter focuses on Wrapper and Embedded methods. Section 4.1 covers Model Criteria metrics. These criteria functions are used to estimate a model's overall quality. The model criterial functions assist Wrapper methods to select important variables, which is discussed in Section 4.2. Section 4.3 discusses Stepwise Selection, which is a classic Wrapper method for variable selection. Section 4.4 discusses Auto Relevance Detection (ARD), an embedded variable selection method. Lastly, Section

4.5 presents a novel voting method that uses a Wrapper method to produce variable rankings similar to Filter methods.

4.1 Model Criteria

There are many different Model Criteria functions that combine a goodness-of-fit objective with a model complexity objective. While each Model Criteria measures model complexity differently, all the functions measure goodness-of-fit using the same criteria, $-2\log(L(\theta))$, where $L(\theta)$ is the maximum likelihood function using θ as the parameter set. Since the preliminary feature selection results (Section 4.6) were generated using Wrapper methods that utilize a Linear Regression Model as the learning system, the general maximum likelihood function should be expressed as follows:

$$L(\theta) = L(Y|\beta, \Sigma) = \frac{1}{2\pi^{k/2}|\Sigma|^{k/2}} e^{-\frac{(Y-X\beta)^{\mathrm{T}}\Sigma^{-1}(Y-X\beta)}{2}}$$

where k is the dimensionality of the multivariate normal (i.e., the number of parameters used in the regression model) and β is a coefficient matrix used to map the input X to a multivariate response Y. However, since our response variable Y is univariate*, the maximum likelihood equation simplifies to the following:

$$L(y|\beta, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{k/2}} e^{-\frac{(y-X\beta)^{\mathrm{T}}(y-X\beta)}{2\sigma^2}}$$

Given that all Model Criteria in this report are applied to univariate Linear Regression Models, one can replace $L(\theta)$ with $L(y|\beta, \sigma^2)$ to frame all Model Criteria for measuring regression complexity.

The first Model Criteria function was defined by Akaike in 1973, called AIC (Akaike's Information Criterion) Akaike (1973). This definition proposed the

* Assuming a univariate response is appropriate, because the response variable used in structure learning via regression methods is univariate in all cases. Additionally, the response variable considered in this chapter is univariate.

evaluation of a model based on the previous likelihood function $L(\theta)$ and a penalty term that attempts to correct the model's bias, under the assumption that the model that best minimizes $\log(L(\theta))$ and minimizes model complexity is the best model. AIC's Criteria function is as follows:

$$\text{AIC}(\theta) = -2\log(L(\theta)) + 2k$$

where k is the number of free parameters that are estimated in the model. After the introduction of AIC, many other Model Criteria functions were introduced, such as Bayesian Information Criterion Schwarz (1978), Minimum Description Length Rissanen (1983), Consistent AIC Bozdogan (1987), and many more. The author of Rumantir (1999) has illustrated that BIC, MDL, CAIC, and many other Model Criteria functions are able to find the true model, if a true model exists, or some approximate parsimonious model, otherwise. However, these methods only compute model complexity in terms of the number of estimated parameters, rather than also including the effect of parameter interactions.

Given that these previous Model Criteria functions compute model complexity without considering parameter interactions, we decided to use the Information Complexity (ICOMP) Bozdogan and Haughton (1998) Criteria. To the best of our knowledge, ICOMP is the only Model Criteria function that measures parameter interaction without the risk of under-fitting the model like CAICF(Bozdogan (1987)). The ICOMP Criteria function is defined as follows:

$$\text{ICOMP(IFIM)} = -2\log(L(\theta)) + 2C(F^{-1}(\theta))$$

where IFIM stands for Inverse Fisher Information Matrix and C is a specified complexity function that maps $F^{-1}(\theta)$, the estimated Inverse Fisher Information Matrix, under the parameters θ, to a single complexity score. Note that lower values of the ICOMP function are preferred. There are many different variants of ICOMP,

each with a different C complexity function and each with a different approximation for $\Sigma(\theta)$ Bozdogan (2003). This proposal focuses on ICOMP(IFIM)$_{\text{Misspec}}$, since it is scale invariant, considers skewness and kurtosis within the model, and helps protect against over-fitting when the $L(\theta)$ function is incorrectly specified Bozdogan (2003). ICOMP(IFIM)$_{\text{Misspec}}$ is defined as follows:

$$\text{ICOMP(IFIM)}_{\text{Misspec}} = -2\log(L(\theta)) + 2C_1(Cov(\theta)_{Misspec})$$

where $Cov(\theta)_{Misspec}$[†] and $C_1(\Sigma)$ are defined as:

$$Cov(\theta)_{Misspec} = F^{-1}RF^{-1}$$

$$C_1(\Sigma) = \frac{p}{2}\log(\frac{tr(\Sigma)}{p}) - \frac{1}{2}|\Sigma|$$

Additionally, Bozdogan (2003) illustrates that when applying ICOMP(IFIM)$_{\text{Misspec}}$ to regression models, F^{-1} and R are defined as:

$$F^{-1} = \begin{bmatrix} \sigma^2(X^{\mathrm{T}}X)^{-1} & 0 \\ 0 & \frac{2\sigma^4}{n} \end{bmatrix}$$

$$R = \begin{bmatrix} \frac{1}{\sigma^4}X^{\mathrm{T}}D^2X & X'1\frac{S_k}{2\sigma^3} \\ (X'1\frac{S_k}{2\sigma^3})' & \frac{(n-q)(K_t-1)}{4\sigma^4} \end{bmatrix}$$

where $D^2 = diag\{\varepsilon_1^2, ..., \varepsilon_n^2\}$ and ε_i^2 is the squared residual error for target y_i, X represents the input data to the regression model, S_k is skewness within the residual errors, and K_t is kurtosis.

[†] Cov$(\theta)_{\text{Misspec}}$ is dependent upon the likelihood function and the learning system, and must be derived for different combinations of the two.

4.2 Genetic Algorithm for Subset Selection

A Genetic Algorithm solves a search problem by considering several candidate solutions in parallel and combining good solutions from the pool of candidate solutions to create new candidate solutions. The hope is that each time the algorithm creates new candidate solutions, they will be superior to the previous candidate solutions. This process is implemented through a set of fairly simplistic, but powerful, operations called selection, crossover, and mutation, which are performed on the current population, or candidate solution set, with respect to a user-defined fitness function that measures solution quality. A candidate solution for our Genetic Algorithm Wrapper for sensor selection is a binary string with a length equal to the number of sensors within the dataset; sensor x_i is included in the solution if the solution has a 1 at index i.

The selection operator determines which candidate solutions will enter the new population without modification and which solutions will be used for constructing new candidate solutions. This process can either uniformly select solutions from the population, select solutions according to a probability distribution derived from each solutions' quality, or select according to a probability distribution defined over the current solution rankings. The latter option is used in this research.

The crossover operation uses the selection operator to pick two candidate solutions from the population and to probabilistically create either one or two candidate solutions. There are many different types of crossover operators; the method used in this research is called *scattered crossover*. This method selects two candidate solutions p_1 and p_2 from the population and generates a random binary string. The new candidate solution copies all elements from p_1 that correspond with a 1 in the binary string and all elements from p_2 that correspond with a 0 in the binary string.

Mutation uses the selection operation to pick a small percentage of the candidate solutions, roughly one or two percent, and then probabilistically determines if it should alter the selected candidate solutions. The alteration is based on a Bernoulli

experiment performed on each binary bit of the selected candidate solutions. This means that with probability p, a single binary bit could change from 1 to 0 or vice versa. There is much controversy over whether or not mutation contributes to finding good candidate solutions, so p is generally set fairly small.

A Genetic Algorithm combines these operators to optimize a fitness function, where the fitness function measures the quality for a candidate solution. In this particular feature selection application, we follow the work presented in Bozdogan (2003), which suggests and illustrates the previously mentioned ICOMP(IFIM) measure as the fitness function, because of its previously stated beneficial properties.

4.3 Stepwise Selection

Stepwise Selection is a greedy search algorithm that attempts to minimize bias by only including variables that contribute statistically significant improvements in performance. This process is carried out iteratively using two passes across the parameter space, where the first pass is a variable inclusion step and the second pass is a variable elimination step. The inclusion pass starts by initializing an initial variable set m, which is generally empty, and iterates over the variable space in a fixed linear order $x_1, x_2, ...x_n$. At each iteration i, the algorithm tests to see if the current model m is statistically worse than the new model m' that includes variable x_i. Model m and model m' are compared using the F-Test to either accept or reject the null hypothesis that variable x_i does not increase model m's performance. If the null hypothesis is rejected with error confidence ρ, then the variable x_i is added to the current model m.

The variable elimination pass starts with model m after completing the inclusion step, and iterates over the variable space in the same fixed linear order. However, at each iteration i, the algorithm tests to see if the current model m is statistically better than model m'' that does not include variable x_i. Model m and model m'' are compared using the same F-Test procedure, but the null hypothesis is now

reformulated as m'' having worse performance than m. If there is not sufficient evidence to reject the null hypothesis with error confidence ρ', then variable x_i is removed from model m.

The inclusion and elimination steps can be repeated until it is either no longer possible to add or remove a variable from the subset, or for a fixed number of iterations if convergence is not possible. In this research, the Stepwise Selection procedure is performed until convergence, with ρ set to five percent and ρ' set to ten percent.

4.4 Auto Relevance Detection

Auto Relevance Detection (ARD) is a Bayesian approach to parameter regularization and estimation. General Bayesian methods use an assumed prior distribution over the model parameters and try to find model parameters that maximize the posterior distribution. That is to say, rather than maximizing the likelihood function $P(X|\theta)$, the Bayesian methods attempts to maximize the posterior distribution $P(X|\theta)P(\theta)$ where $P(\theta)$ is the assumed prior distribution over the model parameters $P(\theta)$. However, ARD methods assume a prior distribution per model parameter, and a a prior distribution over the hyperparameters[‡] for the model parameter priors. According to Tipping (2001), these priors over the hyper parameters can allow individual model parameter distributions to adjust towards a posterior distribution where the MAP estimate is generally zero. Tipping (2001) also states that this result is also based on the data supporting this hypothesis. In other words the sparse model parameters are strictly dependent upon the data. This means that ARD can be used for feature selection, but like the Wrapper methods it does not provide guarantees that the selected features are the true dependent features. That is to say, relevant features will have non-zero weights and irrelevant features will have approximately zero weights.

[‡] The term hyperparameters is used to describe external parameters that are not included in the final model's parameter space.

However, ARD's key disadvantage is that this feature selection attribute may not correspond with variable selection. Li (2007) illustrates this point by noting methods that depend on using model weights for selecting variables are only able to select basis vectors when applied to kernel methods. This means that it may be difficult to use ARD based methods to directly select dependent variables. While it may be possible to address this issue via a transformation between the linear model parameters and the nonlinear model parameters like the work in Li (2007)[§], it is not clear that a transformation will exist between a linear and nonlinear ARD estimated model. However, it is possible to use ARD to select variables directly in other nonlinear models, such as Feed Forward Neural Networks (FFNN). For example, the original ARD work in MacKay et al. (1994) was used to learn regularized FFNNs. While MacKay et al. (1994) did not focus on variable selection, it should be possible to devise a variable selection method using the learned FFNN's weights.

4.5 Feature Ranking

Since we are interested in finding which variables are most useful for building a general prediction model, we framed the problem as a model selection problem. However, each Wrapper method might produce a different best model answer when presented with different subsets of the original dataset. For example, if one uses 75% of a dataset for learning and the remaining 25% for testing purposes, the learning system can provide consistently different best models each time one resamples the data into learning and testing sets. This leaves two options — search for the best model among all possible best models or derive a method to combine the best models seen so far to construct a ranking for each selected variable.

§ The work in Li (2007) proved that it is possible to transform a linear Lasso regression problem into a nonlinear SVM regression problem. This transformation allows the author to use another embedded method (i.e., SVM's sparsity property) to directly select variables via the linear Lasso regression solution. However, this selection method is not based on ARD

Option one is viable, because we are able to use ICOMP(IFIM) to directly compare all seen best models, by selecting the model with the lowest ICOMP(IFIM) score. However, there is an infinite number of best models, and it is not guaranteed that one will find the true best model. As will be seen in Section 4.6, the best model may not always have the smallest ICOMP(IFIM) score, but rather the smallest variance. That is to say, the best sensor subset model will generally have a small variance over a wide range of different learning and testing sets.

Alternatively, one could devise a method for combining each model's best variable subset through voting, since each model is a best model over some set of learning and testing configurations. In our opinion, the voting scheme for combining the best seen subset models should be preferred for models with low ICOMP(IFIM) scores, which have low variance in addition to their low score. Therefore, our voting scheme is defined as follows:

$$v = \frac{\text{ICOMP(IFIM)}_{\text{max}} - \text{ICOMP(IFIM)}_{\text{m}}}{\sigma_m^2}$$

where v is model m's voting power, $\text{ICOMP(IFIM)}_{\text{max}}$ is the score for the worst variable subset in the collection of seen models, and σ_m^2 is model m's ICOMP(IFIM) variance. We then allow each model to cast a positive vote v for each variable present in the model and a negative vote $-v$ for each variable not present in the model. If we sum the votes for each variable, we are able to assign a rank to each variable based on the currently observed best models, by simply sorting all variables final scores in descending order.

4.6 Feature Selection Results

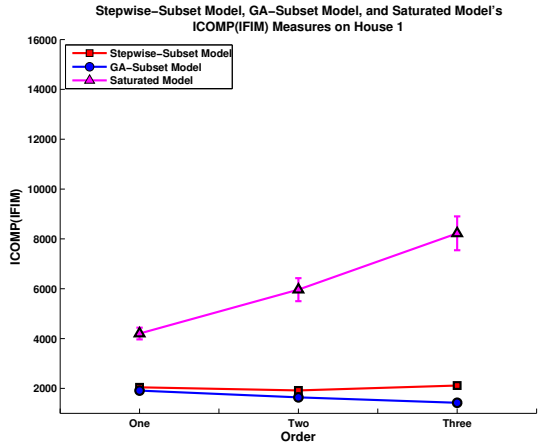
We have organized our feature selection experimental results according to the following order: Campbell Creek House 1, Campbell Creek House 2, Campbell Creek House 3, Across All Houses, Variable Ranking results, and comparisons

against Ground Truth. The individual house sections and the Across All Houses section contain results generated from the selected eight best models. The Variable Ranking section contains results from applying our sensor ranking method mentioned in Section 4.5. The Ground Truth Comparison section presents the results from comparing the best sensors subsets with sizes one through four against the best Markov Order 1 models and the best top 10 sensor sets selected using our ranking method.

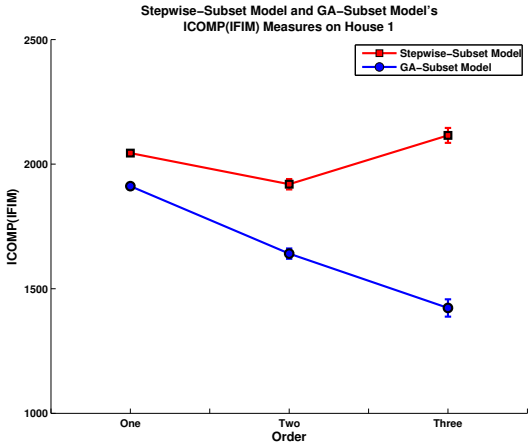
4.6.1 Campbell Creek House 1

Figure 4.1 illustrates the experimental results of comparing the Genetic Algorithm and Stepwise Selection Wrappers based on lowest ICOMP(IFIM) variance, for Campbell Creek House 1. In addition, variables that have missing values were dropped, leaving each method with 87 candidate sensors. Under this particular best model selection, Figure 4.1 shows that the Genetic Algorithm Wrapper finds a more general subset of sensors for Markov Orders 1, 2, and 3. Interestingly, the model selected by the Genetic Algorithm uses more parameters than the model selected with Stepwise Selection for all Markov Orders. The Genetic Algorithm subset uses 57 sensors for Markov Order 1, 69 sensors for Markov Order 2, and 80 sensors for Markov Order 3, while the Stepwise Selection model uses 48 sensors, 58 sensors, and 69 sensors, respectively. This means that the Genetic Algorithm finds sensors it can incorporate without increasing the model complexity, while still producing a slightly better goodness-of-fit as the Stepwise Selection Wrapper (Figure 4.1(e)).

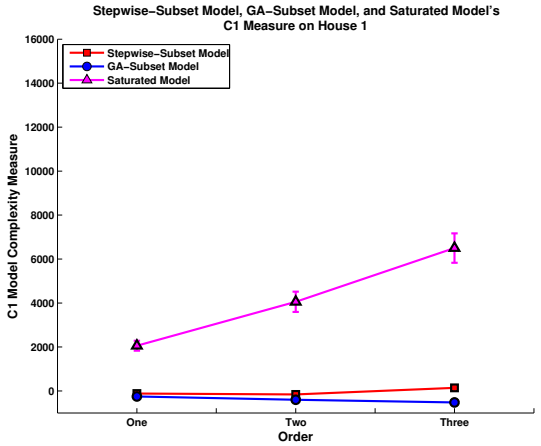
If we change the best model selection policy for the Campbell Creek House 1 dataset with the same 87 candidate sensors to selecting the model with the lowest mean ICOMP(IFIM), then the Genetic Algorithm method shows slight improvement in overall ICOMP(IFIM) criteria, and the Stepwise Selection method's overall ICOMP(IFIM) improves greatly for Orders 2 and 3. However, the goodness-of-fit (Figure 4.2(e)) for Genetic Algorithm methods show improvements over the



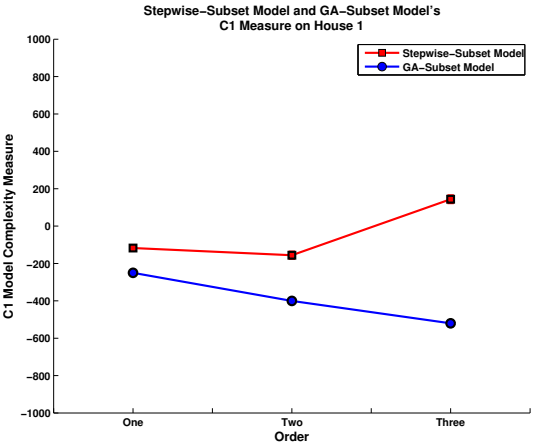
(a) Saturated Model's ICOMP



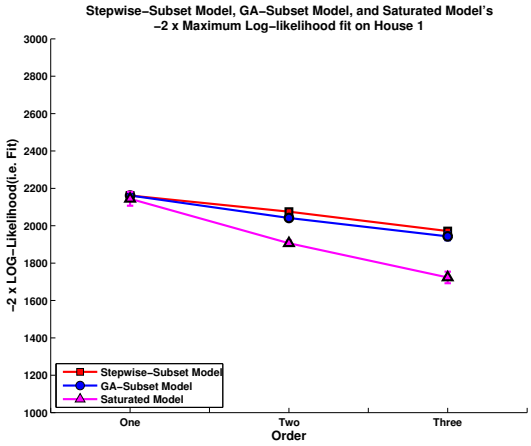
(b) GA and Stepwise Models' ICOMP



(c) Saturated Model's Complexity



(d) GA and Stepwise Model Complexity

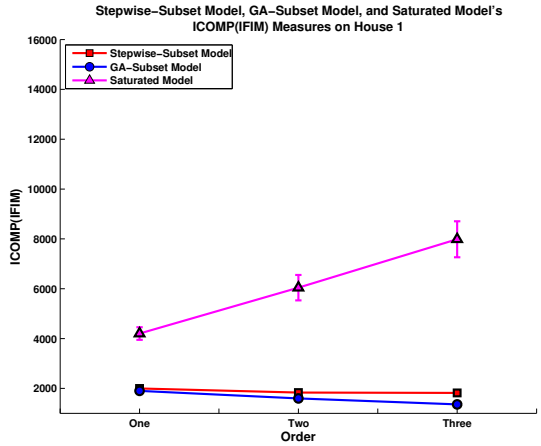


(e) Goodness-of-Fit

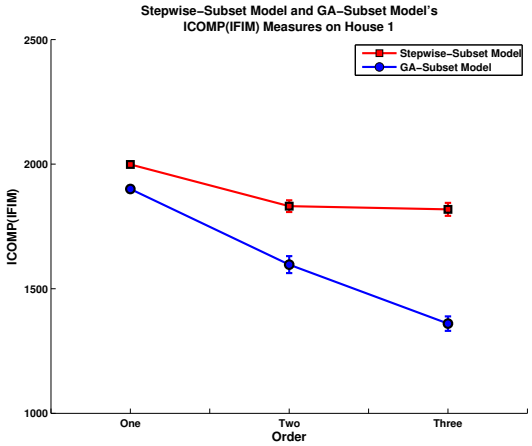
Figure 4.1: These graphs illustrate the experimental results from applying the models with the lowest ICOMP(IFIM) variances on Campbell Creek House 1. Variables with missing data were removed from the dataset for these results.

best variance model (Figure 4.1(e)), while the Stepwise Selection Method shows degradation in performance. It may not be statistically different with a 95% confidence, but the Genetic Algorithm method appears to fit the data better in higher orders. While the overall ICOMP(IFIM) criteria mostly improves, note a slight increase in the overall error range, meaning that these models are possibly more variable than best variance models. This means that when selecting the appropriate sensor subset one needs to consider the possible variance in performance in addition to overall performance. The Stepwise Selection method increases the number of sensors it selects for Markov Orders 1 and 2 in this set of experiments, using 50 sensors, 62 sensors, and 68 sensors. Additionally, the Genetic Algorithm method increases the number of sensors included in Markov Order 1 and 2. It uses 58 sensors, 73 sensors, and 78 sensors for these results.

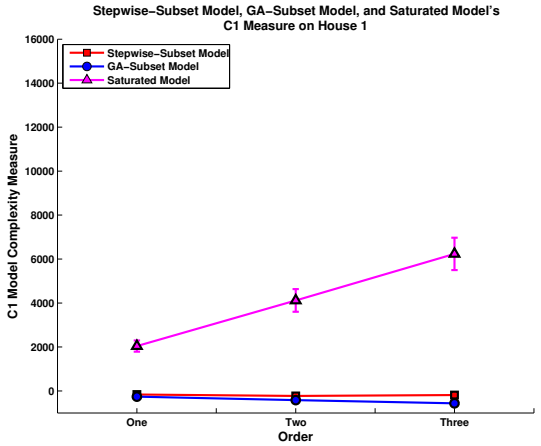
Using the data from the same house, except that missing values are now set to zero and the number of candidate sensors is now 95, Figure 4.3 compares results for the Genetic Algorithm and Stepwise Selection methods based on the model with the lowest ICOMP(IFIM) variance. Under these new conditions, the Genetic Algorithm's ICOMP(IFIM) values are significantly worse than the two models selected when dropping variables with missing values (Figure 4.1(b) and Figure 4.2(b)). Similarly, the Stepwise Selection method's ICOMP(IFIM) values are significantly worse for Markov Orders 1 and 2, but its value for Markov Order 3 is substantially better. It is not quite clear why this Stepwise Selection model performs better than the previous Stepwise Selection models for only Order 3. It could be because the overall fit is better when compared to the previous Stepwise Selection models, and the complexity is higher for order 1 and 2 causing the model to incur an additional penalty for the improved fit. The Genetic Algorithm's fit is significantly worse than the previous lowest ICOMP(IFIM) mean value model (Figure 4.2(b)), yet there is only a slight degradation when compared to the previous lowest variance ICOMP(IFIM) (Figure 4.1(b)). The Genetic Algorithm model used 57 sensors, 73 sensors, and 77 sensors and the Stepwise Selection model used 54 sensors, 66 sensors, and 73 sensors.



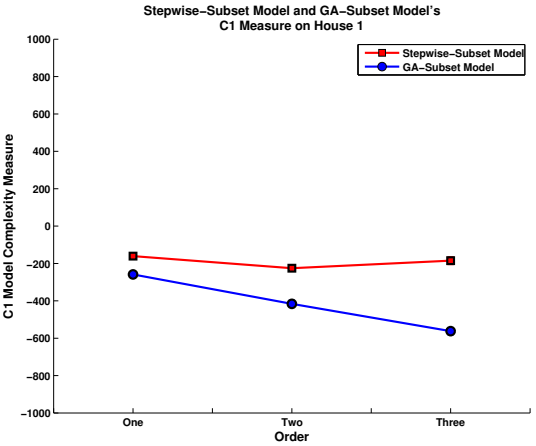
(a) Saturated Model's ICOMP



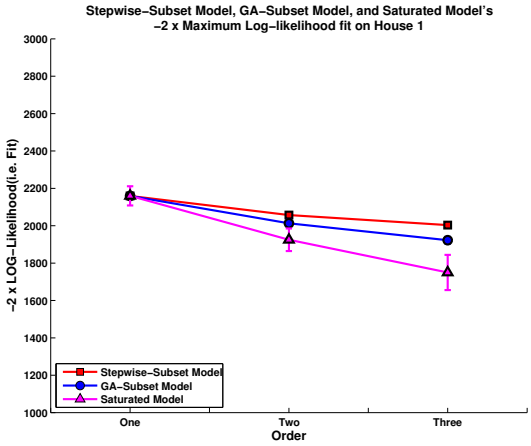
(b) GA and Stepwise Models' ICOMP



(c) Saturated Model's Complexity

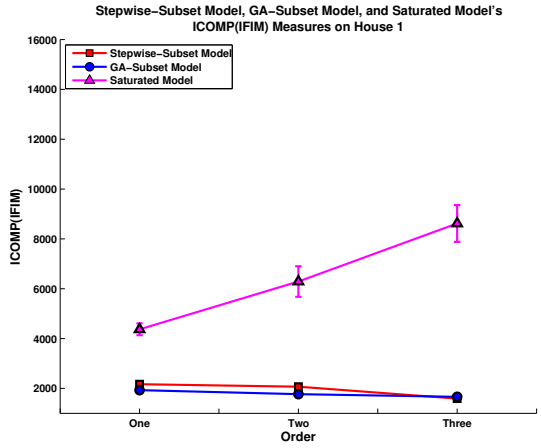


(d) GA and Stepwise Model Complexity

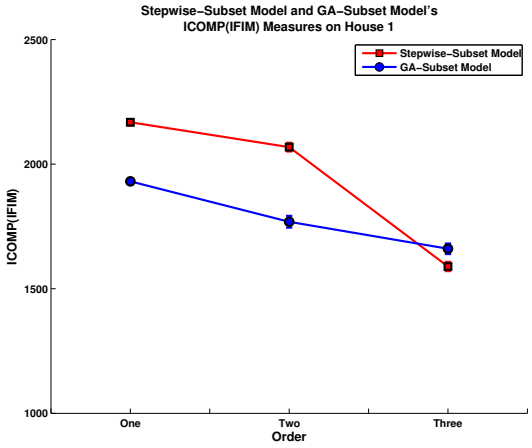


(e) Goodness-of-Fit

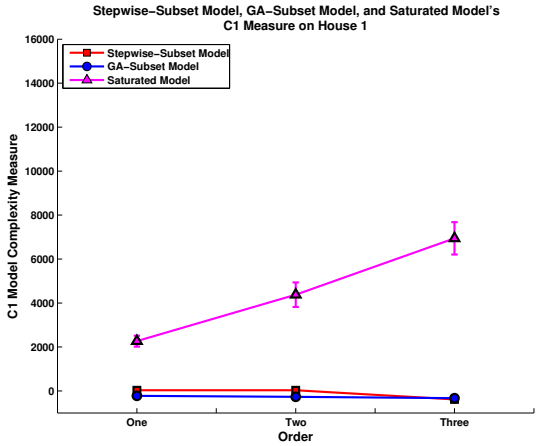
Figure 4.2: These graphs illustrate the experimental results from applying the models with the lowest mean ICOMP(IFIM) on Campbell Creek House 1. Variables with missing data were removed from the dataset for these results.



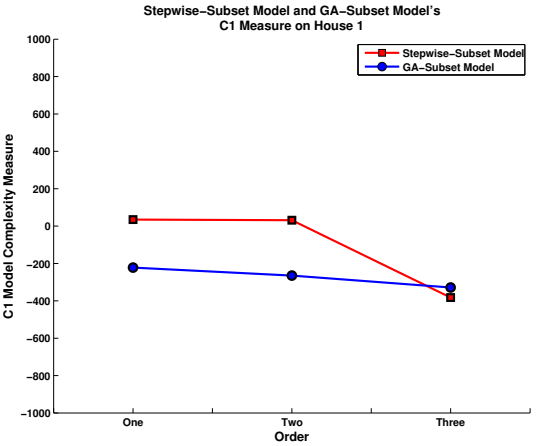
(a) Saturated Model's ICOMP



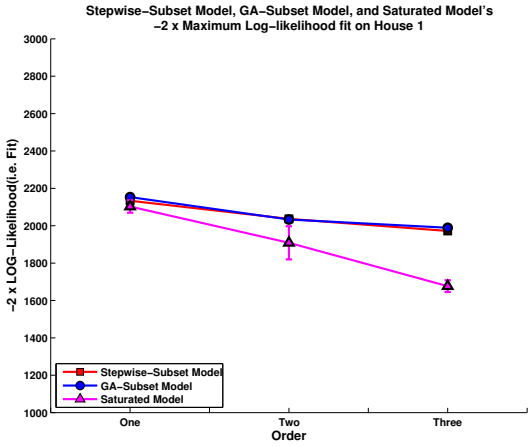
(b) GA and Stepwise Models' ICOMP



(c) Saturated Model's Complexity



(d) GA and Stepwise's Model Complexity



(e) Goodness-of-Fit

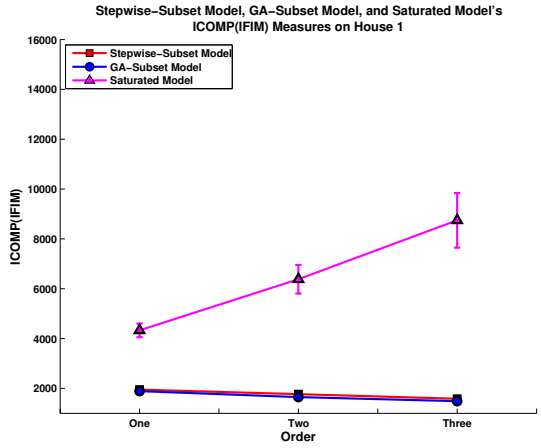
Figure 4.3: These graphs illustrate the experimental results from applying the models with the lowest ICOMP(IFIM) variance on Campbell Creek House 1. All missing values in the data were set to zero for these results.

If we change the best model selection policy for the Campbell Creek House 1 dataset with the same 95 candidate sensors to selecting the model with the lowest mean ICOMP(IFIM), then the Genetic Algorithm model's ICOMP(IFIM) values (Figure 4.4(b)) are much closer to Genetic Algorithm results seen in Figures 4.1(b) and 4.2(b). Also, the Stepwise Selection model's ICOMP(IFIM) values are the best results for this model selection on Campbell Creek House 1. The fit for the Genetic Algorithm model, Figure 4.4(e), is slightly better than the models seen in Figure 4.1(e) and Figure 4.3(e), but is worse than the Genetic Algorithm model in Figure 4.2(e). The Stepwise selection model's fit is mostly identical to the fit seen in Figure 4.1(e). This means the Stepwise Selection model is using sensors that were originally removed from the dataset, and these sensors provide improvement by reducing model complexity. This Stepwise Selection model increased the number of sensors used for Markov Order 1 and 2, compared to the lowest ICOMP(IFIM) variance Stepwise Selection model. It uses 59 sensors, 71 sensors, and 73 sensors, while the Genetic Algorithm model uses 58 sensors, 72 sensors, and 89 sensors.

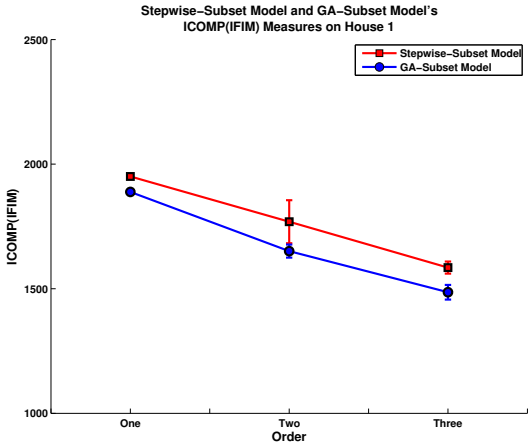
From Figures 4.1, 4.2, 4.3, and 4.4, it is clear that the best Genetic Algorithm Model for Campbell Creek House 1 is the model presented in Figure 4.2, and the best Stepwise Selection Model is presented in Figure 4.4. The dropped variables had a very large impact on the Stepwise Selection method, making it very difficult to find good models under the ICOMP(IFIM) criteria. However, the Genetic Algorithm method in both cases was able to find better models than the Stepwise Selection method, but its best model was found when the variables with missing values were dropped. Ultimately, the Genetic Algorithm method is finding better models than Stepwise Selection on Campbell Creek House 1.

4.6.2 Campbell Creek House 2

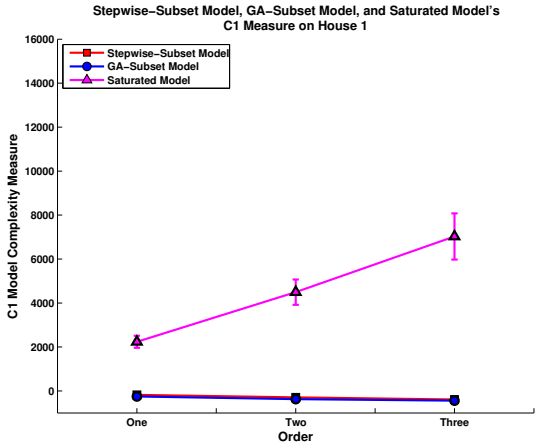
Figure 4.5 compares the results of the Genetic Algorithm and Stepwise Selection Wrappers when selecting the best model, based on lowest ICOMP(IFIM) variance,



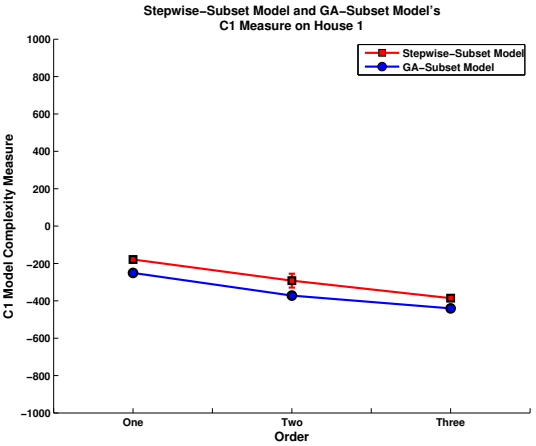
(a) Saturated Model's ICOMP



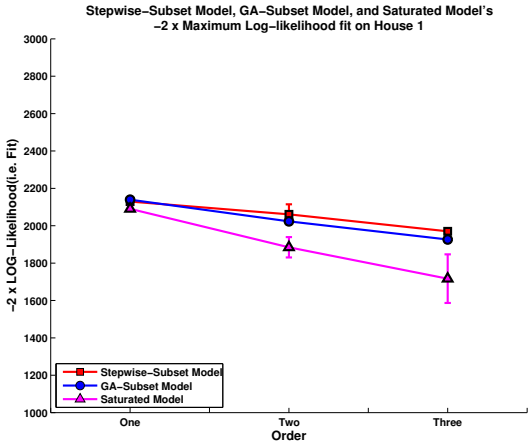
(b) GA and Stepwise Models' ICOMP



(c) Saturated Model's Complexity



(d) GA and Stepwise's Model Complexity



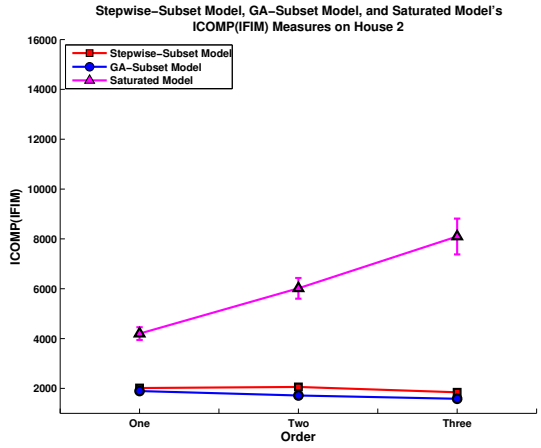
(e) Goodness-of-Fit

Figure 4.4: These graphs illustrate the experimental results from applying the models with the lowest mean ICOMP(IFIM) on Campbell Creek House 1. All missing values in the data were set to zero for these results.

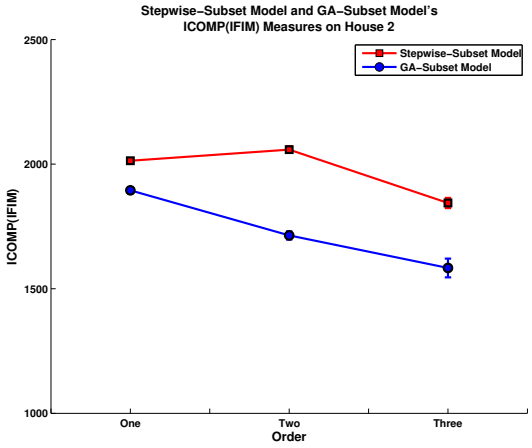
for Campbell Creek House 2. In addition, variables that have missing values were dropped, leaving each method with 84 candidate sensors. Note that under the lowest variance model selection, the Genetic Algorithm finds a better model under the ICOMP(IFIM) metric on this dataset, too. The Genetic Algorithm model uses considerably more sensors for all Markov Orders — 57 sensors, 67 sensors, and 73 sensors, while the Stepwise Selection model uses 46 sensors, 54 sensors, and 53 sensors. The differences in the numbers of sensors explains why the Genetic Algorithm model has a slightly better goodness-of-fit than the Stepwise Selection model (Figure 4.6(e)), because additional sensors included in the model can only increase goodness-of-fit; this is demonstrated with the fully saturated model (Figure 4.6(e)) where a fully saturated model is defined as one that uses all available sensors.

Changing the best model selection strategy to selecting the model with the lowest mean ICOMP(IFIM) increases overall performance on the Campbell Creek House 2 dataset with 84 candidate sensors for the model generated using Stepwise Model Selection (Figure 4.6(b)). The Genetic Algorithm model presents very minor improvements for Markov Order 2 and 3. This stems from the Genetic Algorithm's goodness-of-fit (Figure 4.6(e)) and model complexity (Figure 4.6(d)) not significantly changing because the number of sensors included in the model remains roughly the same as the best variance model. The Genetic Algorithm model in Figure 4.6 uses 60 sensors, 69 sensors, and 73 sensors. Conversely, the Stepwise Selection model's goodness-of-fit appears to increase very slightly. The goodness-of-fit's means are shifted slightly lower than the original means (Figure 4.6(e) and Figure 4.5(e)). The increase stems from the Stepwise Selection method adding additional sensors to the model, using 51 sensors, 62 sensors, and 60 sensors.

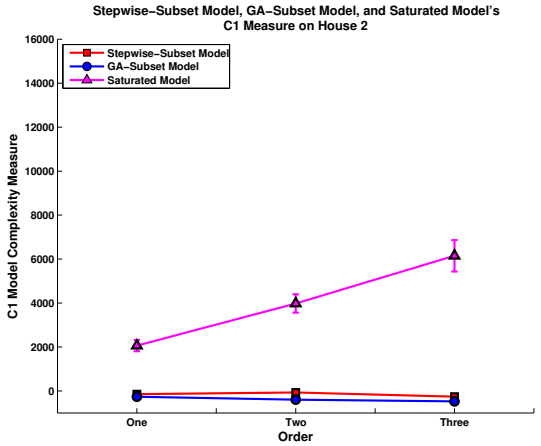
Using the data from the same house, except missing values are now set to zero and the number of candidate sensors is now 103, Figure 4.7 compares results for the Genetic Algorithm and Stepwise Selection methods based on the model with the lowest ICOMP(IFIM) variance. The Genetic Algorithm's overall ICOMP(IFIM) scores show decreases in performance, when compared to results shown in Figure



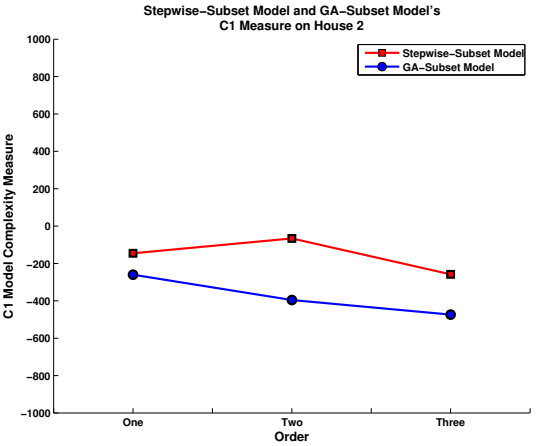
(a) Saturated Model's ICOMP



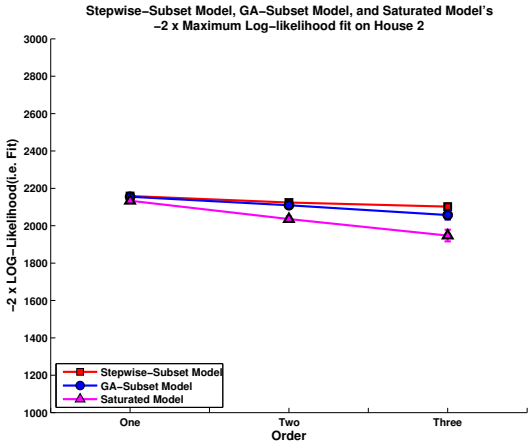
(b) GA and Stepwise Models' ICOMP



(c) Saturated Model's Complexity

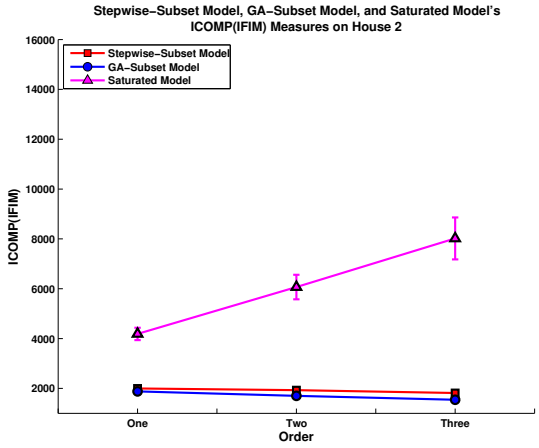


(d) GA and Stepwise's Model Complexity

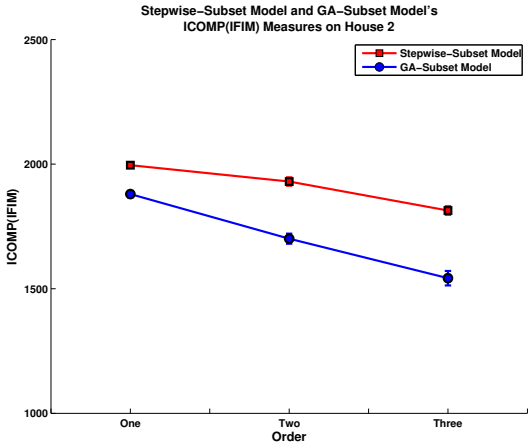


(e) Goodness-of-Fit

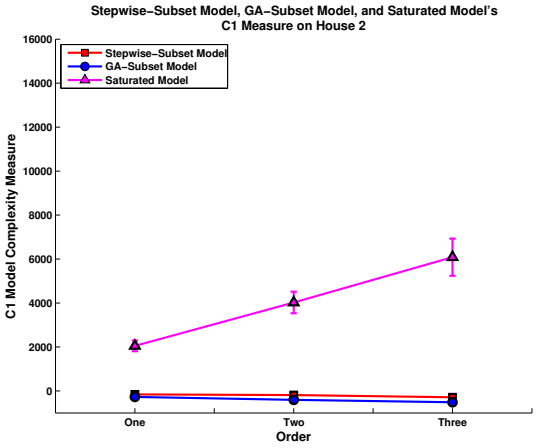
Figure 4.5: These graphs illustrate the experimental results from applying the models with the lowest ICOMP(IFIM) variances on Campbell Creek House 2. Variables with missing data were removed from the dataset for these results.



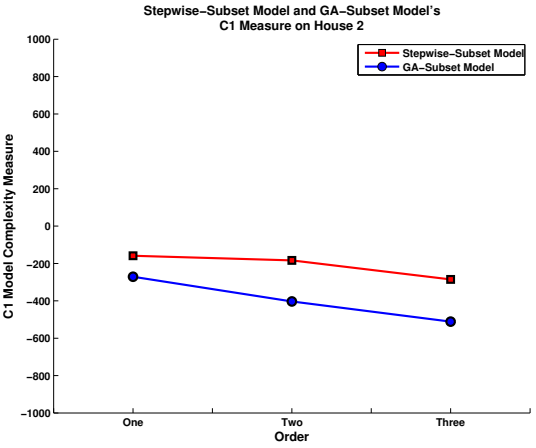
(a) Saturated Model's ICOMP



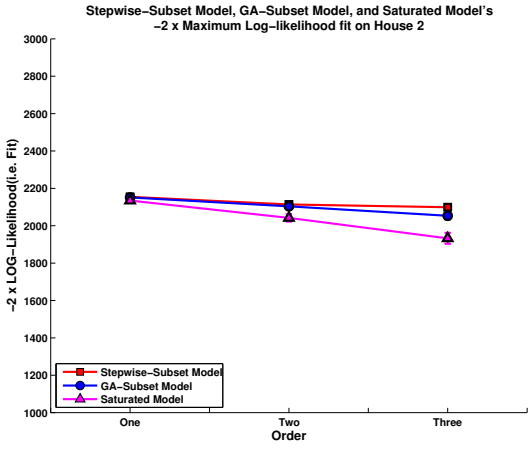
(b) GA and Stepwise Models' ICOMP



(c) Saturated Model's Complexity



(d) GA and Stepwise's Model Complexity



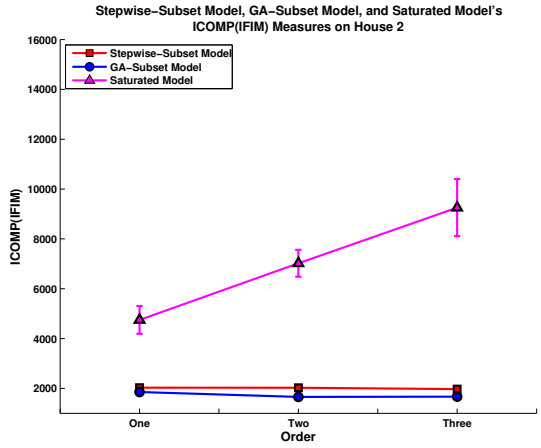
(e) Goodness-of-Fit

Figure 4.6: These graphs illustrate the experimental results from applying the models with the lowest mean ICOMP(IFIM) on Campbell Creek House 2. Variables with missing data were removed from the dataset for these results.

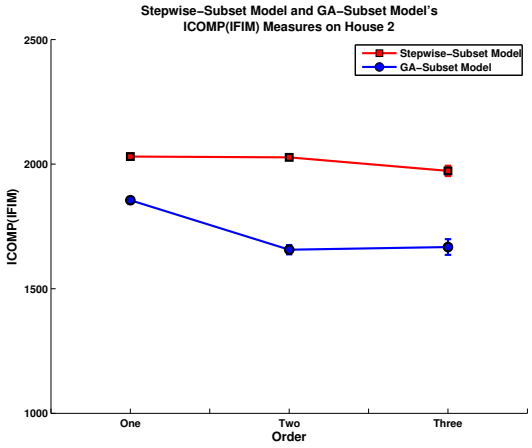
4.6(b) and Figure 4.5(b). The overall ICOMP(IFIM) scores for Stepwise Selection are slightly better than the results seen in Figure 4.5(b), but are considerably worse than the results shown in Figure 4.6(b). Additionally, the goodness of fit is slightly worse for both the Genetic Algorithm and Stepwise Selection compared to the previous models. The Genetic Algorithm uses 67 sensors for Markov Order 1, 85 sensors for Markov Order 2, and 91 sensors for Markov Order 3, while the Stepwise Selection method uses 63 sensors, 62 sensors, and 67 sensors, respectively.

Changing the best model selection strategy to selecting the model with the lowest mean ICOMP(IFIM) increases overall performance on the Campbell Creek House 2 dataset with 103 candidate sensors. The models generated using Stepwise Model Selection for Markov Orders 1 and 3 (Figure 4.8(b)) perform better than all other Stepwise Models on Campbell Creek House 2. However, its performance for Order 2 remains essentially the same as all other Stepwise Models. Additionally, the Genetic Algorithm method finds the best performing model in terms of ICOMP(IFIM), compared to the other models presented in Figures 4.6(b), 4.5(b), and 4.7(b). The Genetic Algorithm method uses 69 sensors, 78 sensors, and 93 sensors. The best performing Stepwise Selection method uses 61 sensors, 62 sensors, and 68 sensors.

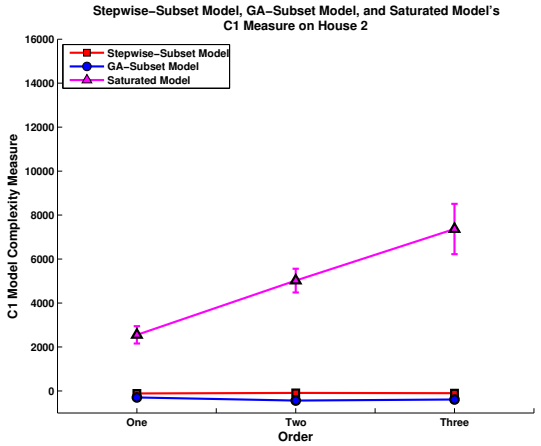
From Figures 4.5, 4.6, 4.7, and 4.8, it is clear that the best Genetic Algorithm Model for Campbell Creek House 2 is the model presented in Figure 4.8, and the best Stepwise Selection Model is presented in Figure 4.8. Similar to House 1, the dropped variables had a very large impact on the Stepwise Selection method, making it very difficult to find good models under the ICOMP(IFIM) criteria. However, the Genetic Algorithm method in both cases was able to find better models than the Stepwise Selection method, but its best model was found when the variables with missing values were dropped. Ultimately, the Genetic Algorithm method is finding better models than Stepwise Selection on Campbell Creek House 2.



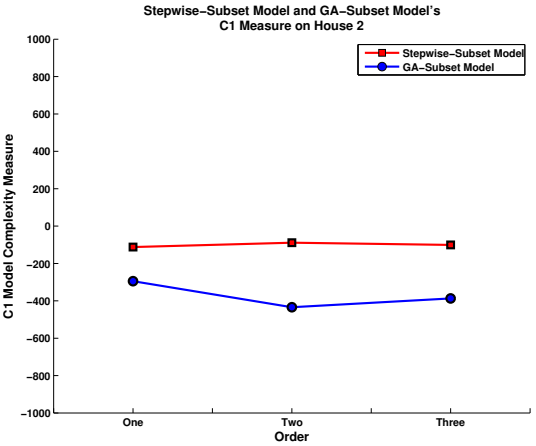
(a) Saturated Model's ICOMP



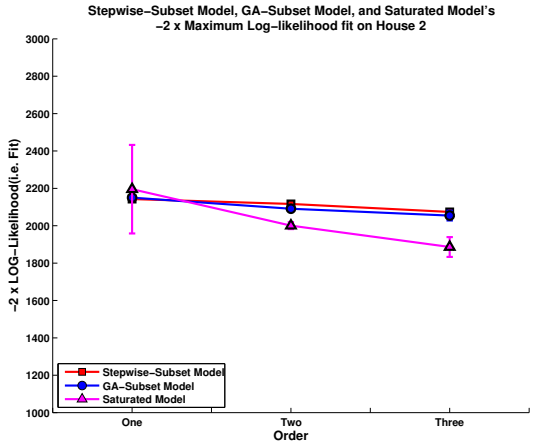
(b) GA and Stepwise Models' ICOMP



(c) Saturated Model's Complexity

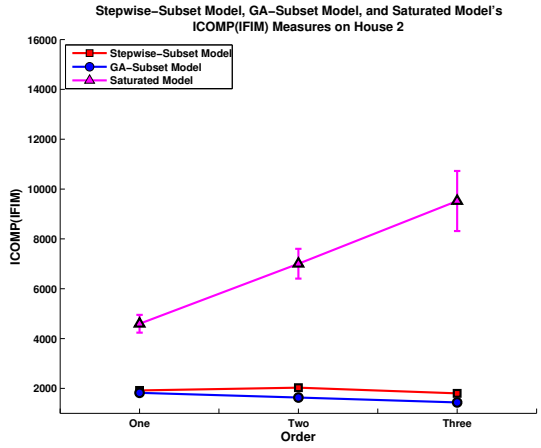


(d) GA and Stepwise Models' Complexity

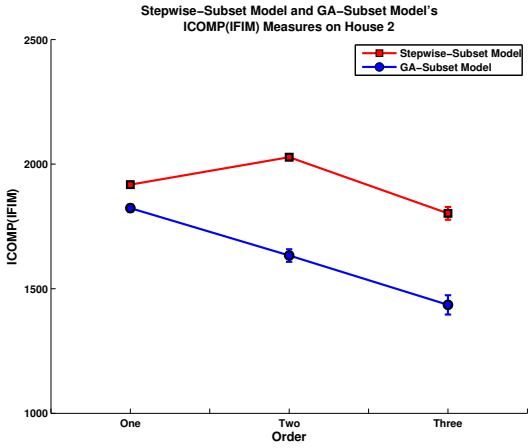


(e) Goodness-of-Fit

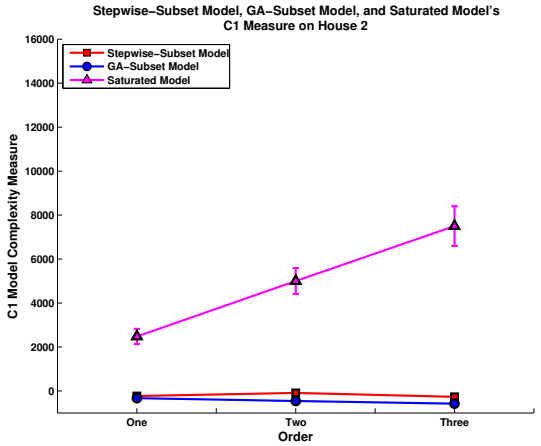
Figure 4.7: These graphs illustrate the experimental results from applying the models with the lowest ICOMP(IFIM) variance on Campbell Creek House 2. All missing values in the data were set to zero for these results.



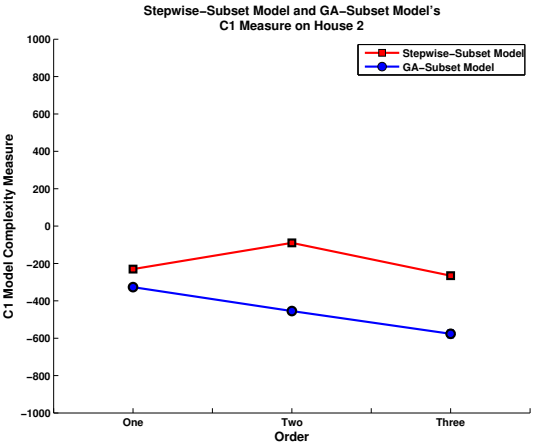
(a) Saturated Model's ICOMP



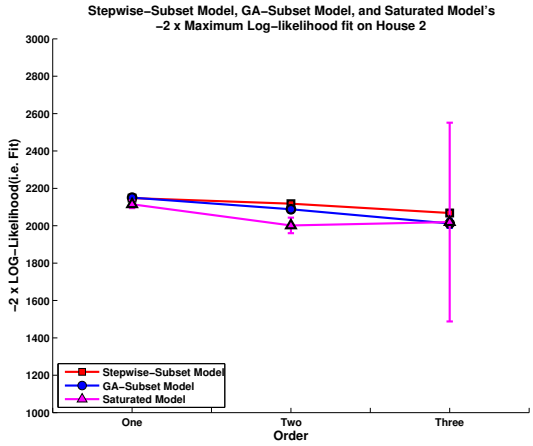
(b) GA and Stepwise Models' ICOMP



(c) Saturated Model's Complexity



(d) GA and Stepwise Models' Complexity



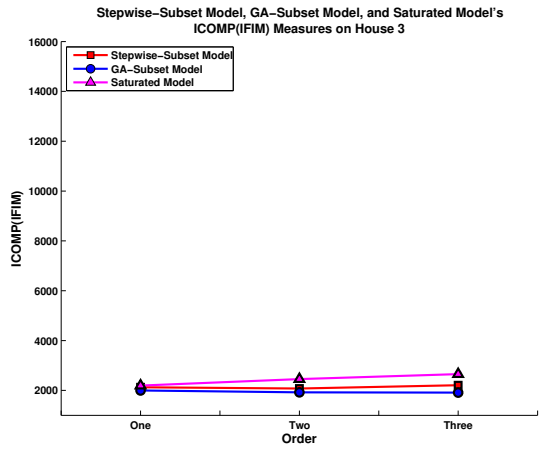
(e) Goodness-of-Fit

Figure 4.8: These graphs illustrate the experimental results from applying the models with the lowest mean ICOMP(IFIM) on Campbell Creek House 2. All missing values in the data were set to zero for these results.

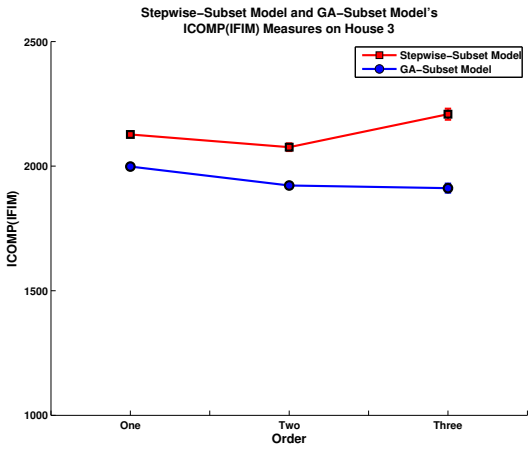
4.6.3 Campbell Creek House 3

Figure 4.9 compares the results of the Genetic Algorithm and Stepwise Selection Wrappers when selecting the best model based on lowest ICOMP(IFIM) variance on Campbell Creek House 3. In addition, variables that have missing values were dropped, leaving each method with 77 candidate sensors. Recall that House 3 is the house for which a linear regression technique is not able to obtain a near-perfect mapping from x_t to y_t, while these mappings were successfully found for Houses 1 and 2. With this in mind, note that the ICOMP(IFIM) scores are considerably higher (and thus worse) compared to the ones seen for Houses 1 and 2. Additionally, for all Markov Orders, the model selected with the Genetic Algorithm is better in terms of ICOMP(IFIM) and model complexity (Figure 4.9(b) and Figure 4.9(d)), but the goodness-of-fit is essentially the same as the Stepwise Selection model (Figure 4.9(e)). The model selected by the Genetic Algorithm uses 41 sensors, 48 sensors, and 56 sensors, while the model selected by Stepwise Selection uses 49 sensors, 52 sensors, and 53 sensors.

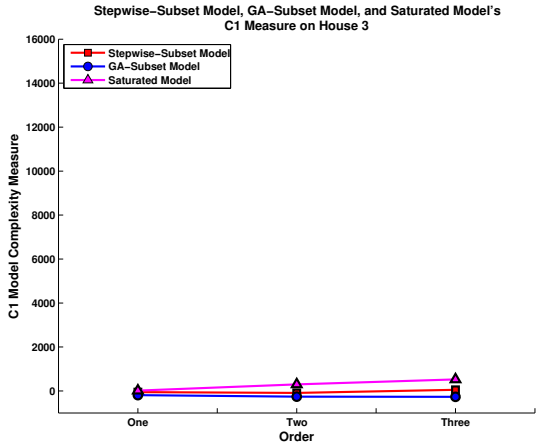
Changing the best model selection strategy to one of selecting the model with the lowest mean ICOMP(IFIM) value on the House 3 dataset, with 77 candidate sensors, shows improvement for Markov Order 3 in term of ICOMP(IFIM) values for both methods, but little to no increase for goodness-of-fit. Figures 4.9 and 4.10 strongly suggest that a different approach is required for modeling House 3, because the overall model complexity for the fully saturated model is extremely low (Figure 4.9(c)) when compared to the overall model complexity for the fully saturated model on Houses 1 and 2 (Figure 4.1(c) and Figure 4.7(c)). This argues that there are complex nonlinear relationships between House 3's sensor data and the actual energy consumption; we currently believe this difference stems from the fact that House 3 has the capability to produce a portion of its own electricity using solar panels. However, one can clearly see that the Stepwise Selection and Genetic Algorithm methods still minimize the



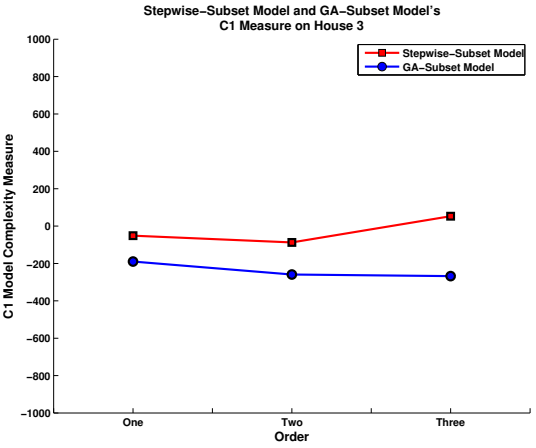
(a) Saturated Model's ICOMP



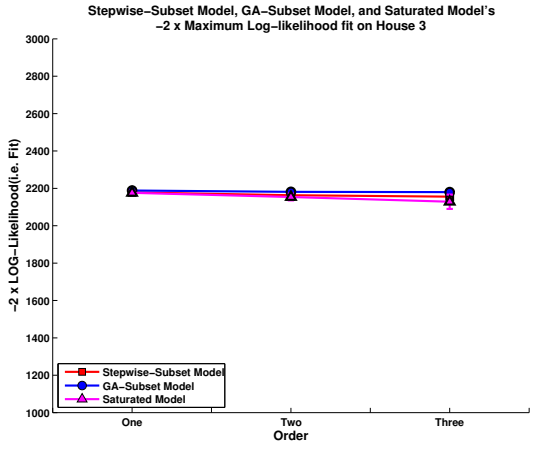
(b) GA and Stepwise Models' ICOMP



(c) Saturated Model's Complexity



(d) GA and Stepwise Models' Complexity



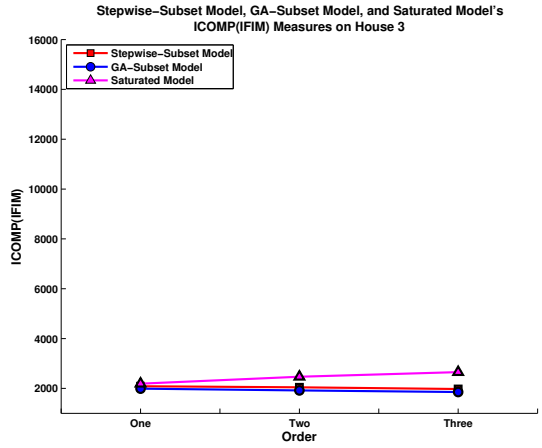
(e) Goodness-of-Fit

Figure 4.9: These graphs illustrate the experimental results from applying the models with the lowest ICOMP(IFIM) variances on Campbell Creek House 3. Variables with missing data were removed from the dataset for these results.

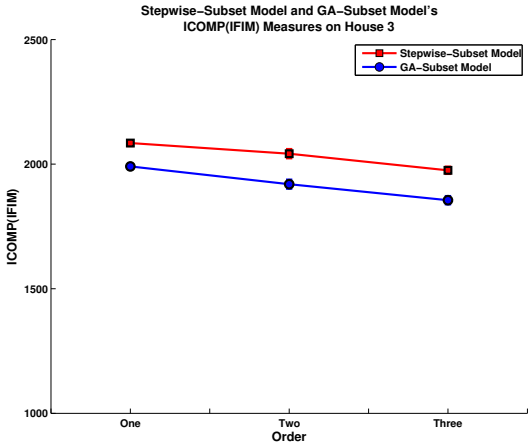
selected model complexity, even though a Linear Regression Model may not be the most appropriate Learning method.

Using the data from the same house, except missing values are now set to zero and the number of candidate sensors is 128, Figure 4.11 compares results for the Genetic Algorithm and Stepwise Selection methods based on the model with the lowest ICOMP(IFIM) variance. Comparing the ICOMP(IFIM) values from the Genetic Algorithm and Stepwise Selection models (Figure 4.11(b)) against previous ICOMP(IFIM) values (Figure 4.10(b) and Figure 4.9(b)), one will see that there is considerable degradation in the Genetic Algorithm's performance, while the Stepwise Selection is showing increases in performance for all orders. However, the model generated by Stepwise Selection for Markov Order 3 has a fairly large standard deviation, implying the model is highly variable and unstable. In addition, one should notice that both methods are more than likely over-fitting or under-fitting the training examples as the Markov Order increases, which is clearly visible from the decreasing performance in the goodness-of-fit (Figure 4.11(e)). The Genetic Algorithm uses 63 sensors, 93 sensors, and 109 sensors, while Stepwise Selection uses 71 sensors, 91 sensors, and 75 sensors.

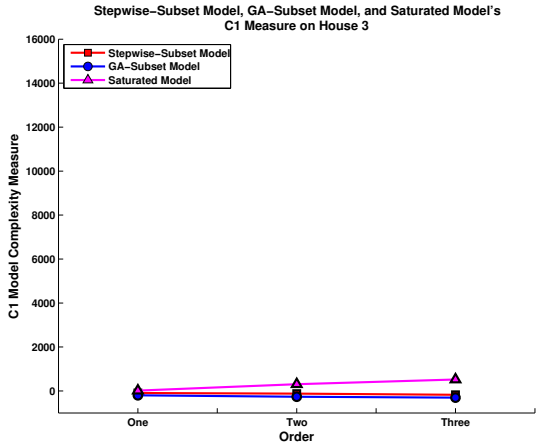
Changing the best model selection strategy to selecting the model with the lowest mean ICOMP(IFIM) increases overall performance on the Campbell Creek House 3 dataset with 128 candidate sensors for all models generated by both methods (Figure 4.12(b)). However, Stepwise Selection has a slightly better goodness-of-fit for orders 2 and 3 compared to the Genetic Algorithm (Figure 4.12(e)), but Stepwise Selection's model complexity is much higher than the model complexity for the Genetic Algorithm for order 2. Yet, both methods have equivalent complexity for Markov Order 3, making the Stepwise Selection model the best model compared to the previous models in Figure 4.11(e), Figure 4.9(e), and Figure 4.10(e). In addition, the Stepwise Selection method uses 76 sensors, 86 sensors, and 85 sensors, while the Genetic Algorithm method uses 77 sensors, 88 sensors, and 107 sensors. This implies



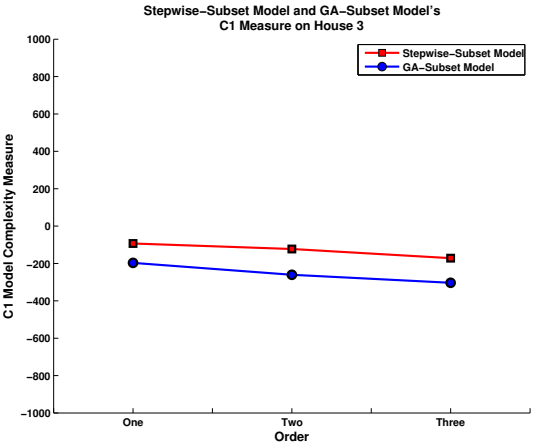
(a) Saturated Model's ICOMP



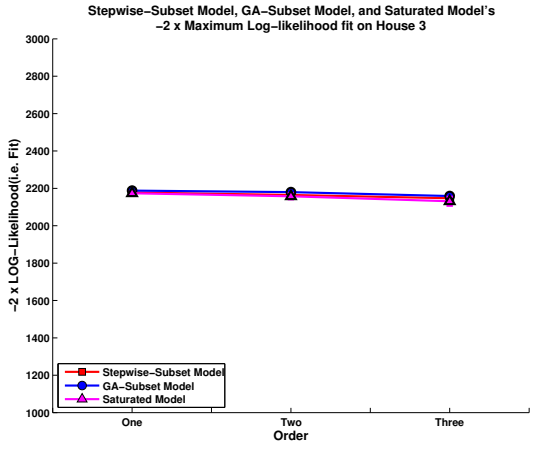
(b) GA and Stepwise Models' ICOMP



(c) Saturated Model's Complexity

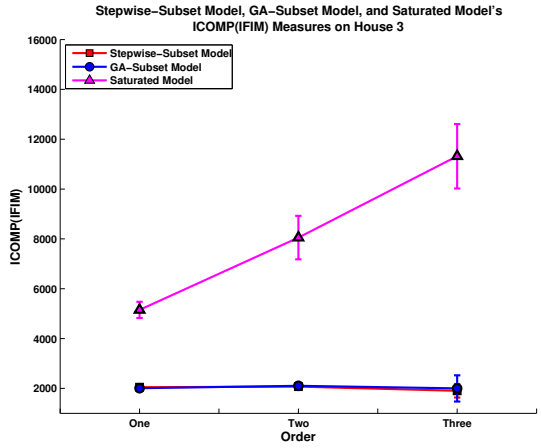


(d) GA and Stepwise Models' Complexity

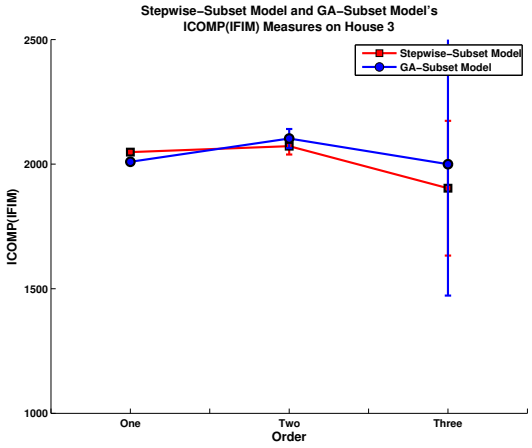


(e) Goodness-of-Fit

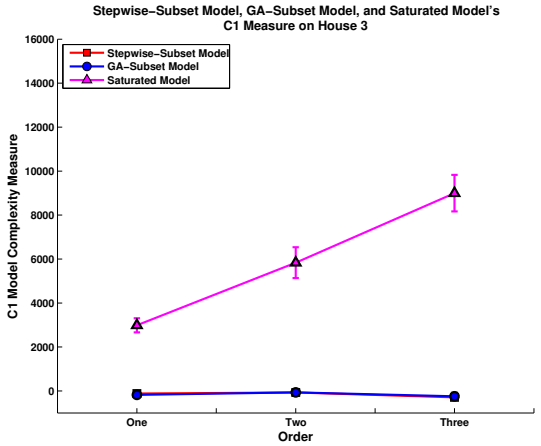
Figure 4.10: These graphs illustrate the experimental results from applying the models with the lowest mean ICOMP(IFIM) on Campbell Creek House 3. Variables with missing data were removed from the dataset for these results.



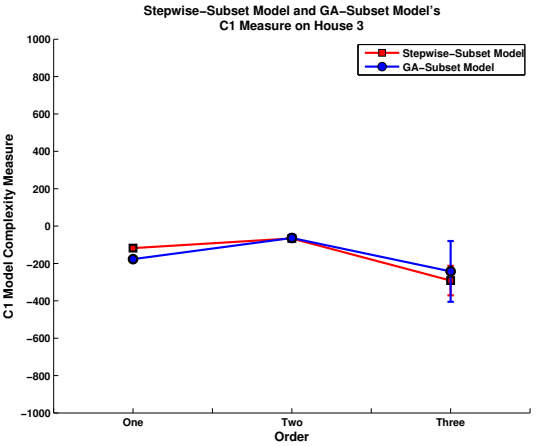
(a) Saturated Model's ICOMP



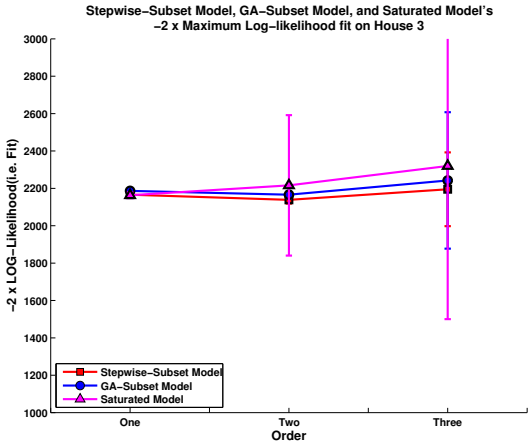
(b) GA and Stepwise Models' ICOMP



(c) Saturated Model's Complexity



(d) GA and Stepwise Models' Complexity



(e) Goodness-of-Fit

Figure 4.11: These graphs illustrate the experimental results from applying the models with the lowest ICOMP(IFIM) variance on Campbell Creek House 3. All missing values in the data were set to zero for these results.

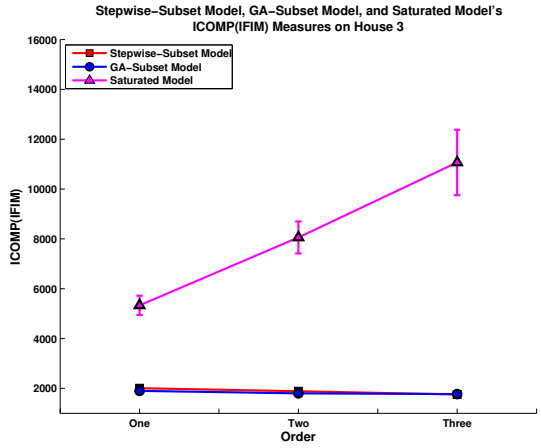
that the model generated from using the Genetic Algorithm in Figure 4.11 is over-fitting for higher Markov Orders, and the model generated from Stepwise Selection, also shown in Figure 4.11, is under-fitting for Markov Order 3 and over-fitting for Markov Order 2.

From Figures 4.9, 4.10, 4.11, and 4.12 it is clear that the best Genetic Algorithm Model and Stepwise Selection Model for House 3 are the models presented in Figure 4.12. Additionally, we observe, yet again, that dropping variables with missing values had a significant impact on the Stepwise Selection method, and setting missing values to zero showed impact on the Genetic Algorithm method. While the Genetic Algorithm is for the most part producing better models on this data set, Stepwise Selection produced the best model, Markov Order 3 model in Figure 4.12, making it the better choice for this particular dataset.

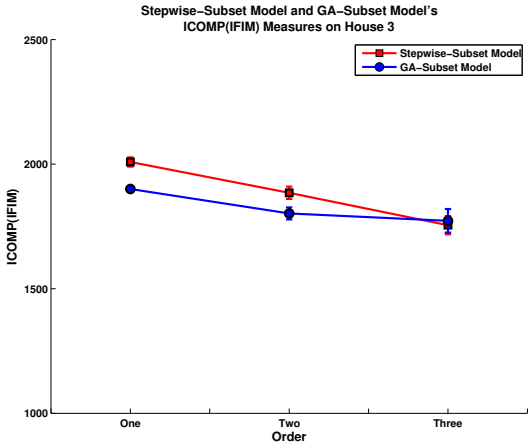
4.6.4 Across All Houses

Figure 4.13 compares the results of the Genetic Algorithm and Stepwise Selection Wrappers when selecting the best model based on lowest ICOMP(IFIM) variance, across all Campbell Creek Houses. In addition, variables that have missing values were dropped, leaving each method with 75 candidate sensors. According to the ICOMP(IFIM) values in Figure 4.13(a), the Genetic Algorithm is generating better models than Stepwise Selection for all Markov Orders. The goodness-of-fit is equivalent for all models generated with each method (Figure 4.13(e)), implying that the Genetic Algorithm is consistently minimizing model complexity and maintaining goodness-of-fit. The Genetic Algorithm is using 50 sensors, 61 sensors, and 69 sensors, while Stepwise Selection is using 56 sensors, 63 sensors, and 62 sensors.

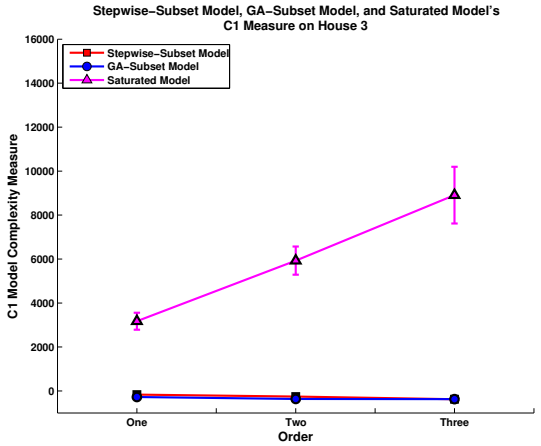
Changing the best model selection strategy to one of selecting the model with the lowest mean ICOMP(IFIM) value across all houses, with 75 candidate sensors, one will see that the Genetic Algorithm's ICOMP(IFIM) values in Figure 4.14(b) indicate no changes in performance quality. However, comparing the Stepwise Selection results



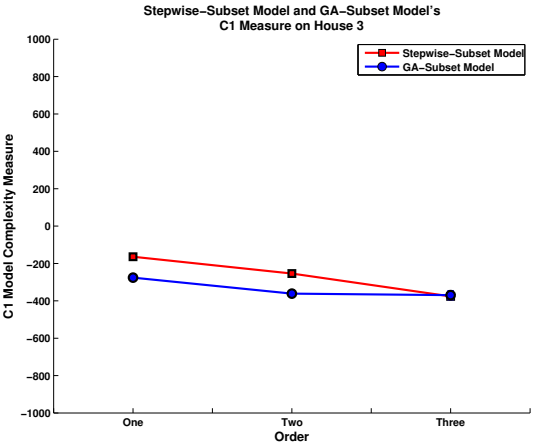
(a) Saturated Model's ICOMP



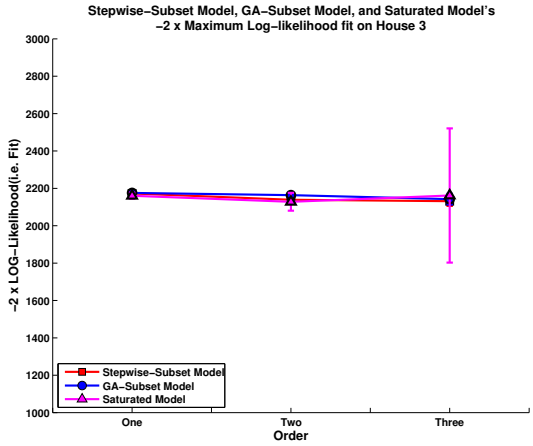
(b) GA and Stepwise Models' ICOMP



(c) Saturated Model's Complexity

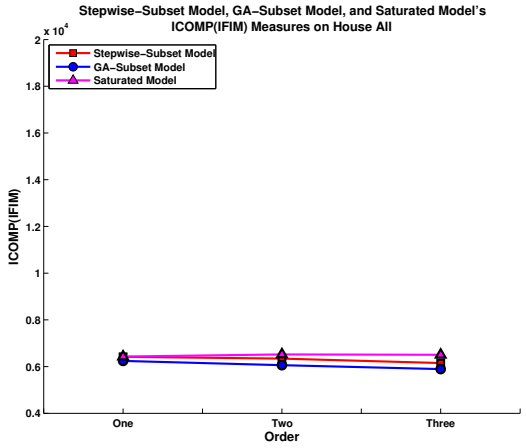


(d) GA and Stepwise Models' Complexity

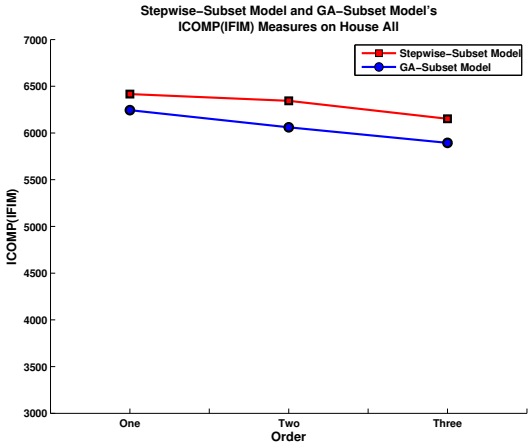


(e) Goodness-of-Fit

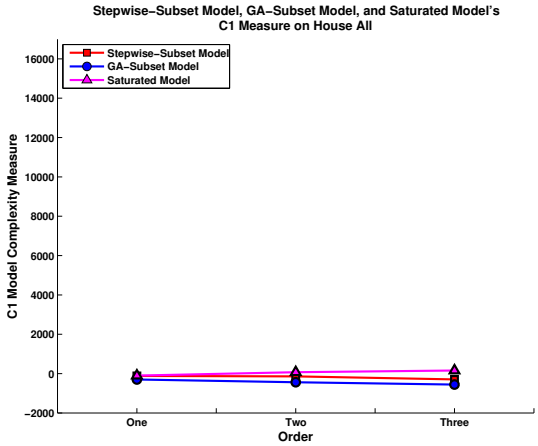
Figure 4.12: These graphs illustrate the experimental results from applying the models with the lowest mean ICOMP(IFIM) on Campbell Creek House 3. All missing values in the data were set to zero for these results.



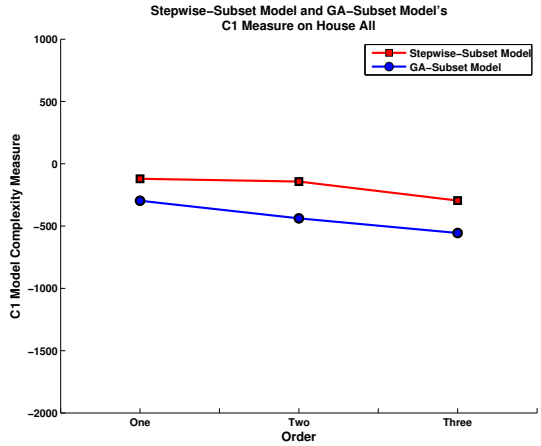
(a) Saturated Model's ICOMP



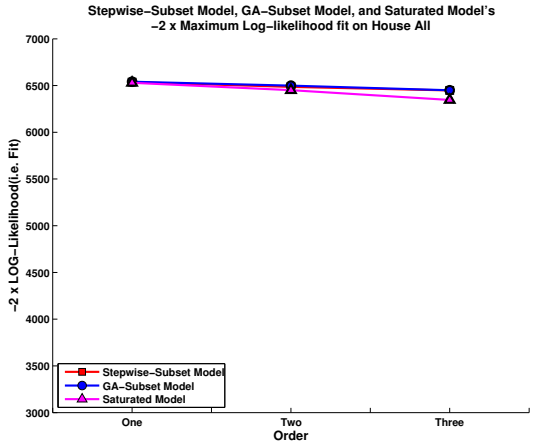
(b) GA and Stepwise Models' ICOMP



(c) Saturated Model's Complexity



(d) GA and Stepwise Models' Complexity



(e) Goodness-of-Fit

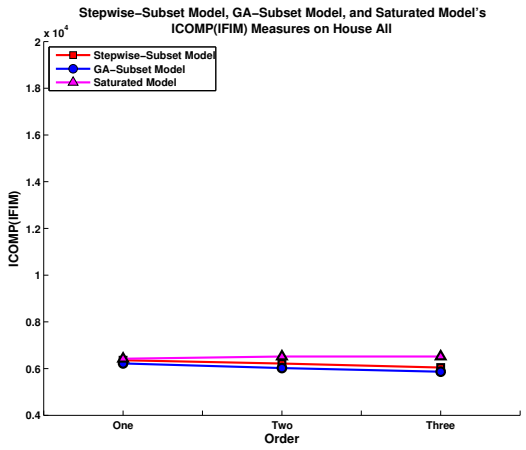
Figure 4.13: These graphs illustrate the experimental results from applying the models with the lowest ICOMP(IFIM) variance across all houses. Variables with missing data were removed from the dataset for these results.

in the same Figure to the results in Figure 4.13(b), one sees a slight increase in performance for Markov Order 2, and the results for Markov Order 1 and 3 are about the same. In addition, the goodness-of-fits for these models (Figure 4.14(e)) are identical to the goodness-of-fits observed in Figure 4.13(e). The Genetic Algorithm is using 62 sensors, 62 sensors, and 68 sensors, while Stepwise Selection is using 55 sensors, 61 sensors, and 67 sensors.

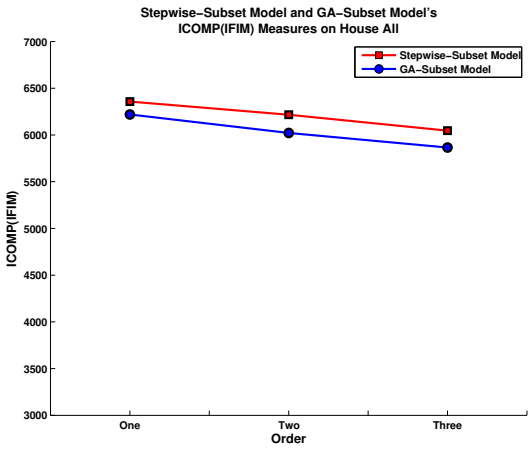
Using the same data, except missing values are now set to zero and the number of candidate sensors is 141, Figure 4.15 compares results for the Genetic Algorithm and Stepwise Selection methods based on the model with the lowest ICOMP(IFIM) variance. Figure 4.15(b) shows that the Genetic Algorithm and Stepwise Selection methods' ICOMP(IFIM) values have a very large increase in performance compared to previous results in Figure 4.13(b) and Figure 4.14(b). The increase performance mainly stems from both methods showing decreases in model complexity (Figure 4.15(d)), but there are slight improvements in goodness-of-fit (Figure 4.15(e)) as well. The Genetic Algorithm uses 93 sensors, 123 sensors, and 118 sensors, while Stepwise Selection uses 98 sensors, 107 sensors, and 109 sensors.

Changing the best model selection strategy to selecting the model with the lowest mean ICOMP(IFIM) increases overall performance across all houses with 141 candidate sensors for all models generated by both methods (Figure 4.16(b)). Comparing the model complexity for both models in Figure 4.16(d) with all previous models on this data set, one will see that these models obtain the lowest complexity for Markov Orders 2 and 3, and the same model complexity as the models seen in Figure 4.15(d), for Markov Order 1. Additionally, the goodness-of-fit (Figure 4.16(e)) for these models is essentially the same as the goodness-of-fit presented for the models seen in Figure 4.15(e). This best performing Genetic Algorithm algorithm model uses 95 sensors, 123 sensors, and 124 sensors, while the best performing Stepwise Selection uses 85 sensors, 104 sensors, and 110 sensors.

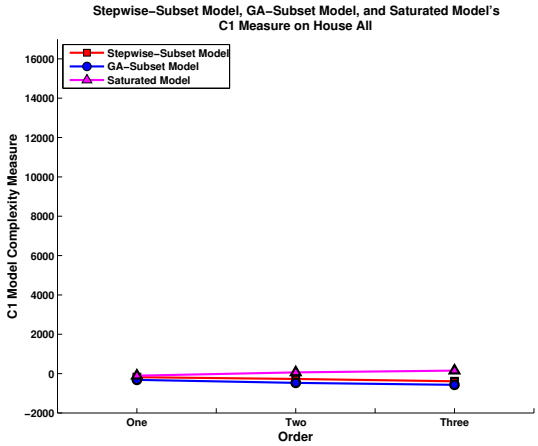
From Figures 4.13, 4.14, 4.15, and 4.16 it is clear that the best Genetic Algorithm Model and Stepwise Selection Model across all houses are presented in Figure 4.16.



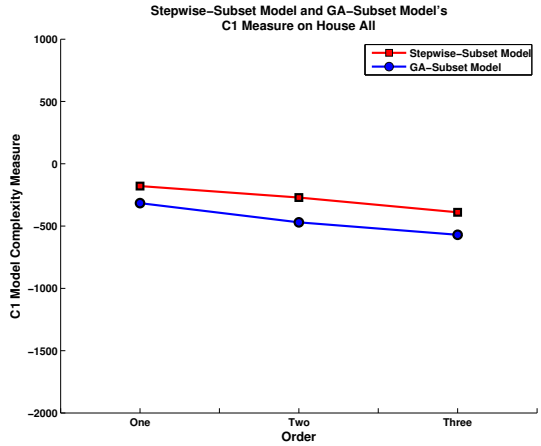
(a) Saturated Model's ICOMP



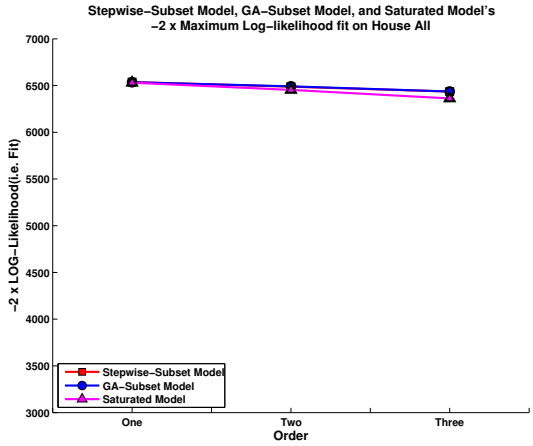
(b) GA and Stepwise Model's ICOMP



(c) Saturated Model's Complexity

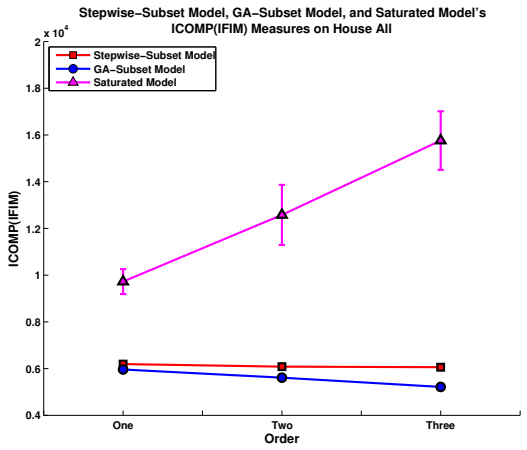


(d) GA and Stepwise Model's Complexity

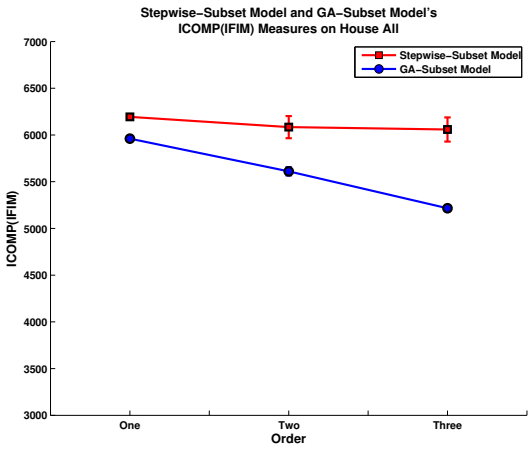


(e) Goodness-of-Fit

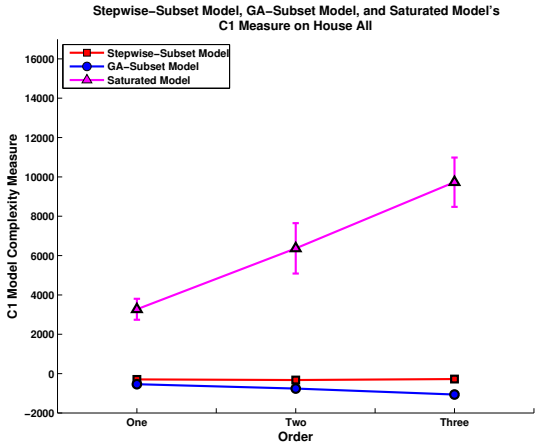
Figure 4.14: These graphs illustrate the experimental results from applying the models with the lowest mean ICOMP(IFIM) across all houses. Variables with missing data were removed from the dataset for these results.



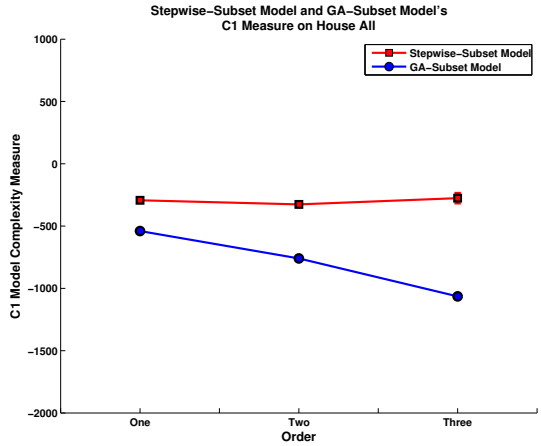
(a) Saturated Model's ICOMP



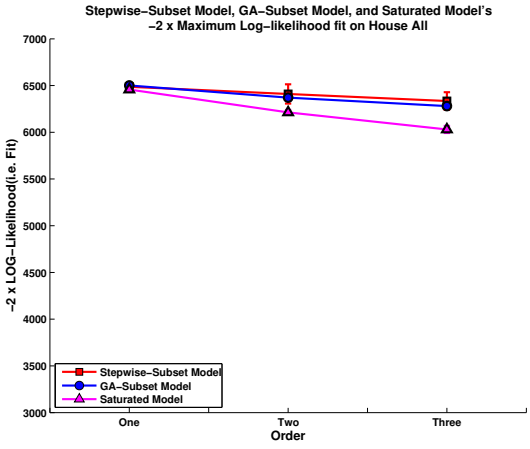
(b) GA and Stepwise Models' ICOMP



(c) Saturated Model's Complexity

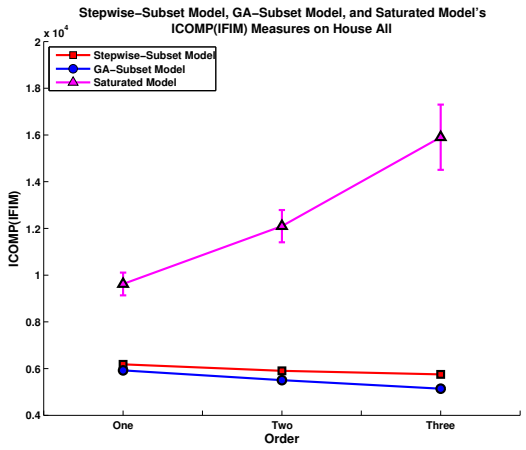


(d) GA and Stepwise Models' Complexity

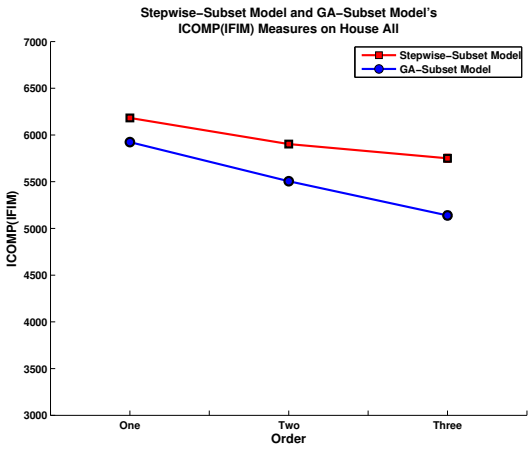


(e) Goodness-of-Fit

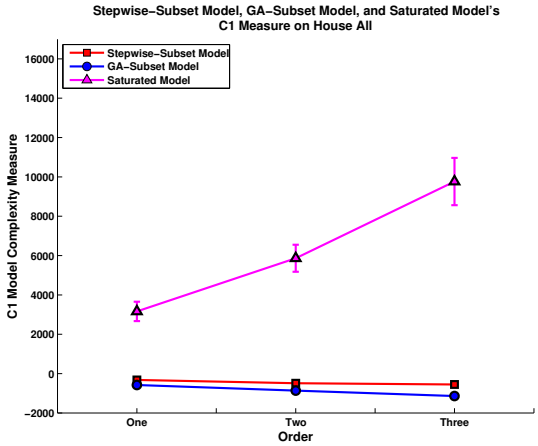
Figure 4.15: These graphs illustrate the experimental results from applying the models with the lowest ICOMP(IFIM) variance across all houses. All missing values in the data were set to zero for these results.



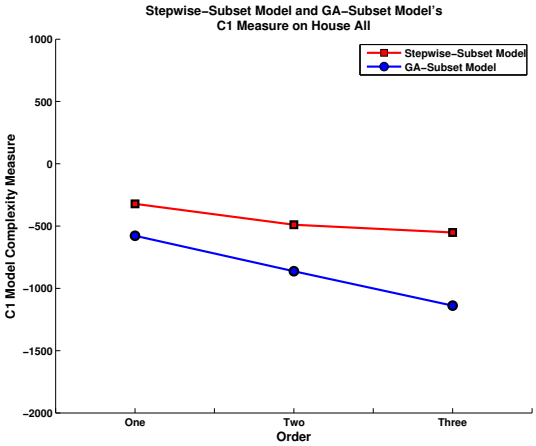
(a) Saturated Model's ICOMP



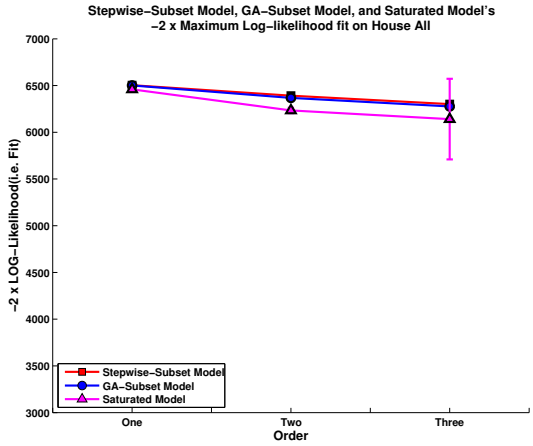
(b) GA and Stepwise Models' ICOMP



(c) Saturated Model's Complexity



(d) GA and Stepwise Models' Complexity



(e) Goodness-of-Fit

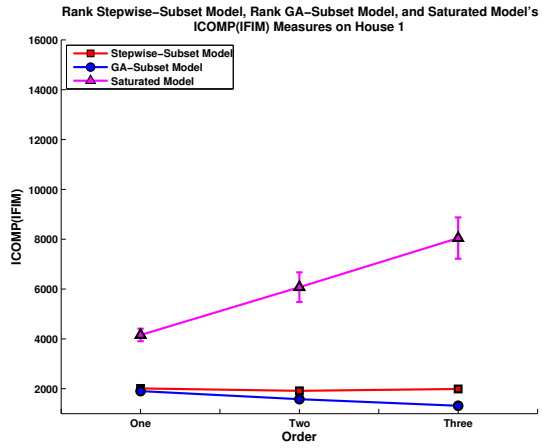
Figure 4.16: These graphs illustrate the experimental results from applying the models with the lowest mean ICOMP(IFIM) across all houses. All missing values in the data were set to zero for these results.

In the previously presented results, Stepwise Selection was generally the only method significantly affected by dropping variables with missing values; however, the Genetic Algorithm method was greatly affected as well on this data set. The key reason for this change is due to the fact that not all the houses have the same sensors, and dropping sensors with missing values greatly limits the number of available sensors, which greatly restricts the Genetic Algorithm's search space.

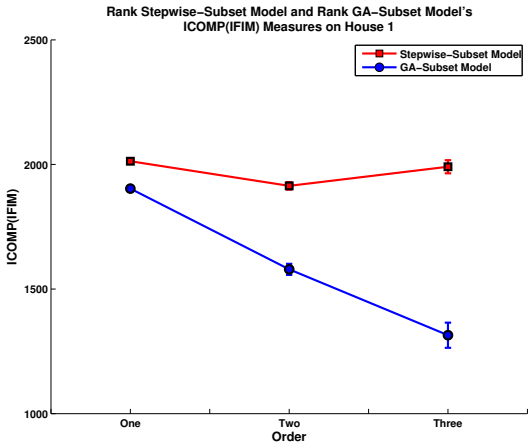
4.6.5 Variable Ranking

Figures 4.17, 4.18, 4.19, and 4.20 present the results from applying our sensor ranking technique to determine the best model, when variables with missing values were removed. Recall that the sensor ranking method combines all best models found for each method, and then selects the top k sensors from the list to use in the final model. For all of these results, we heuristically set k equal to the number of sensors whose total vote is greater than zero. Additionally, Tables 4.1 and 4.2 show the top ten sensors for both methods on Markov Order 1.

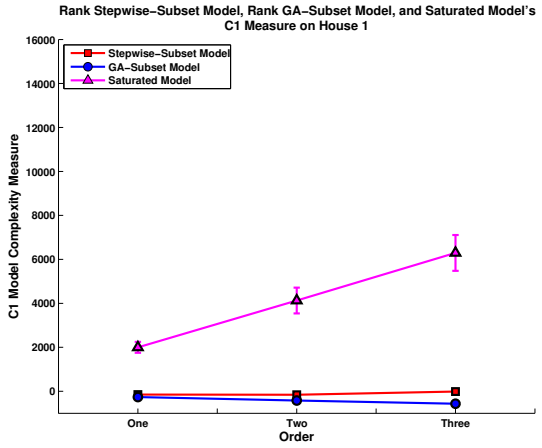
Comparing the results from Figure 4.17 with the previous results for Campbell Creek House 1 (Figures 4.1, 4.2, 4.3, and 4.4), one can see that the Rank Model created from combining all the models generated by the Genetic Algorithm is better than the previously seen best models on House 1 (Figure 4.4). However, the Rank model constructed from the Stepwise Selection models in Figure 4.17 is worse than the previously seen best Stepwise Selection model on House 1 (Figure 4.4), but is better than the model seen in Figure 4.1 for all Markov Orders, and is better than the model in Figure 4.3 for Markov Order 1 and 2. Combining Stepwise models, where variables with missing values were removed, gives some improvement in performance, but most likely the removed variables are contributing to the poor performance. This Stepwise Rank Model is created using models that have previously demonstrated poor performance, because the variables with missing data were removed. This means one cannot expect a large performance increase when the base models are poor.



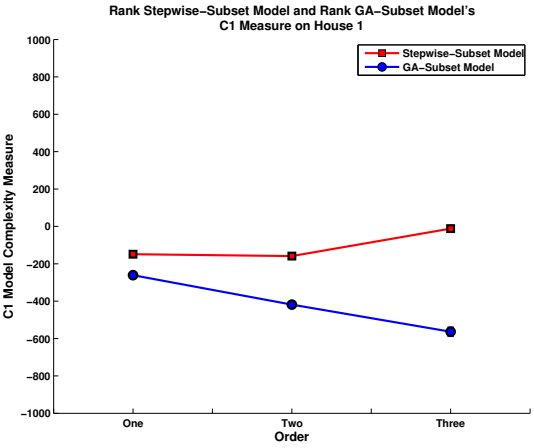
(a) Saturated Model's ICOMP



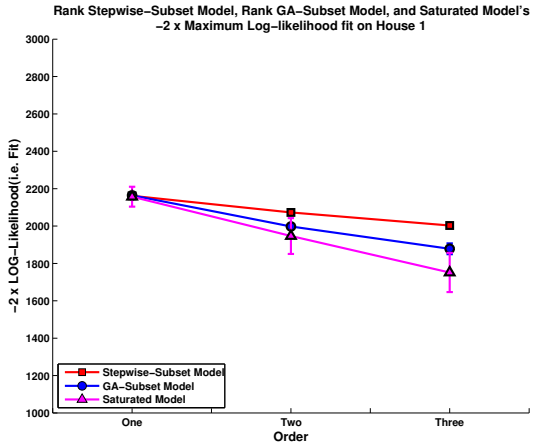
(b) GA and Stepwise Models' ICOMP



(c) Saturated Model's Complexity



(d) GA and Stepwise Models' Complexity

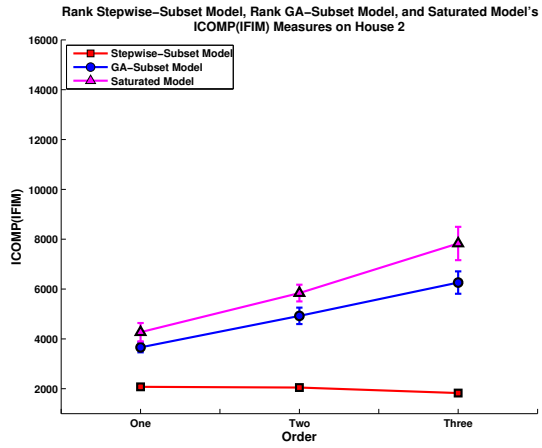


(e) Goodness-of-Fit

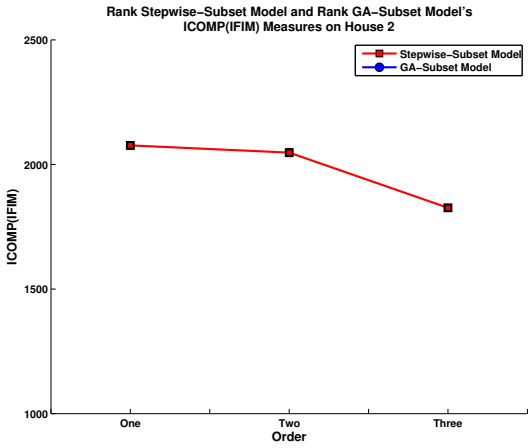
Figure 4.17: Experimental results for Campbell Creek House 1's Rank Models with dropped variables that have missing data.

On Campbell Creek House 2, the Rank Model created from combining the Genetic Algorithm subset models compared to all previously presented models (Figures 4.5, 4.6, 4.7, and 4.8) is the worst model (Figure 4.18). The model's complexity is fairly close to the fully Saturated Model (Figure 4.22(c)) for all Markov Orders, making it much more undesirable than the previously presented models. The Rank Model constructed from the Stepwise Selection models performs much better than the Rank Genetic Algorithm model, but is not better than the best model seen in the previously presented House 2 results. The Stepwise Rank Model's Markov Order 3 (Figure 4.18(a)) has better performance than the Stepwise Selection model for the Markov Order 3 seen in Figure 4.6(a), but worse performance on Markov Order 1 and 2. Additionally, comparing the same rank model to the Stepwise Selection results in Figure 4.5(a), we observe that Rank Model's Markov Order 1 performance is worse, but the performance is better for the other Markov Orders. Comparing the Stepwise Rank Model's result against the Stepwise selected models without removed variables, we see that previous results in Figure 4.8 are better for all Markov Orders, and the results in Figure 4.7 are better for Markov Order 1 and 2.

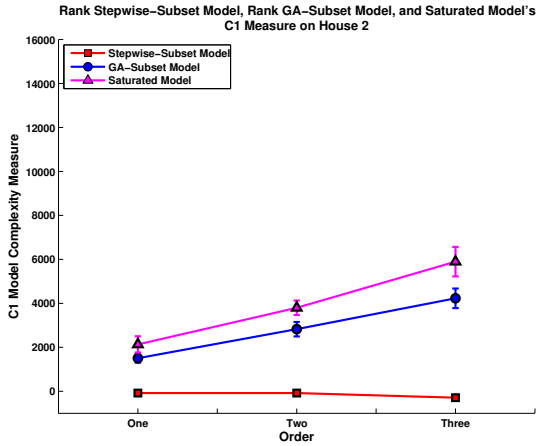
Comparing the Rank Model created from the Genetic Algorithm for House 3 (Figure 4.19), where variables with missing values were dropped, against all previously presented results for House 3 (Figures 4.9, 4.10, 4.11, and 4.12), one can see that the rank model for Markov Order 2 and Markov Order 3 has better performance than the Genetic Algorithm results in Figure 4.10, and Markov Order 1 performance is the same. In addition, the Genetic Algorithm results in Figure 4.9 are worse than the Genetic Rank Model for Markov Orders 2 and 3, but the same for Markov Order 1. Comparing the Genetic Rank Model results against the Genetic Algorithm results in Figure 4.12, one will see that the Genetic Rank Model is worse for all Markov Orders. Yet, the Genetic Rank Model produces better results than the Genetic Algorithm results presented in Figure 4.11. The Stepwise Rank Model results are generally similar to the previous Stepwise Selection results on House 3, and only present better performance when compared against the poorer performing Stepwise models.



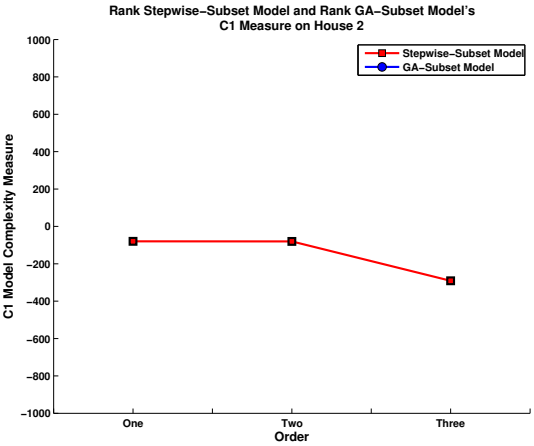
(a) Saturated and GA Models' ICOMP



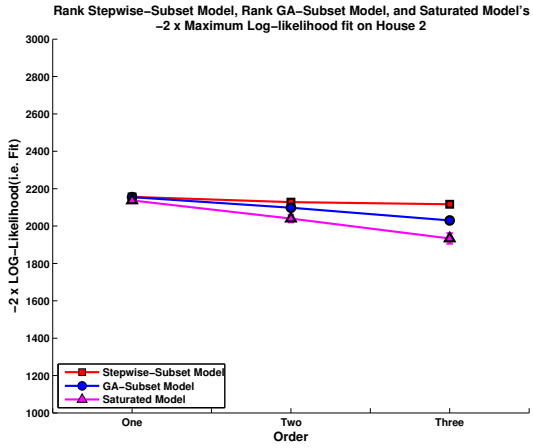
(b) Stepwise Model's ICOMP



(c) Saturated and GA Models' Complexity

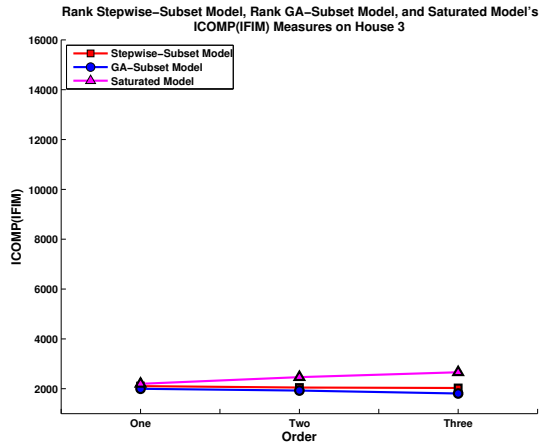


(d) Stepwise Model Complexity

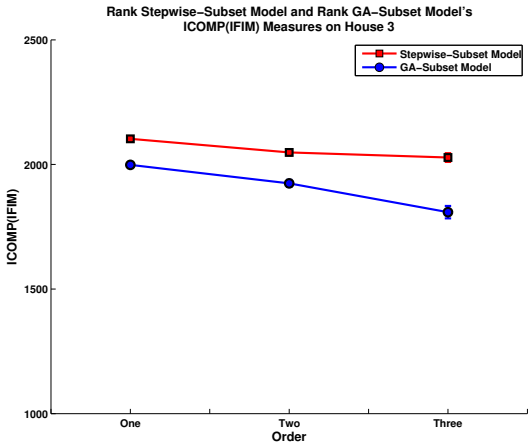


(e) Goodness-of-Fit

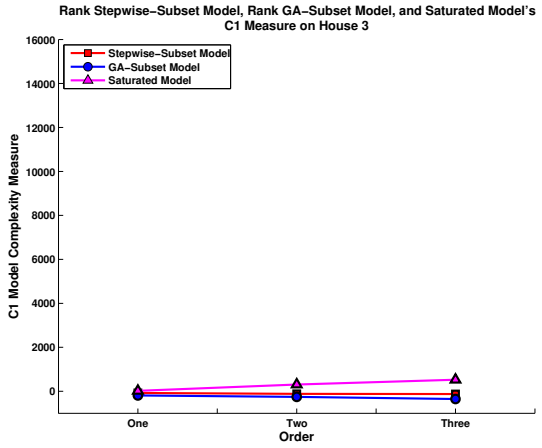
Figure 4.18: Experimental results for Campbell Creek House 2's Rank Models with dropped variables that have missing data.



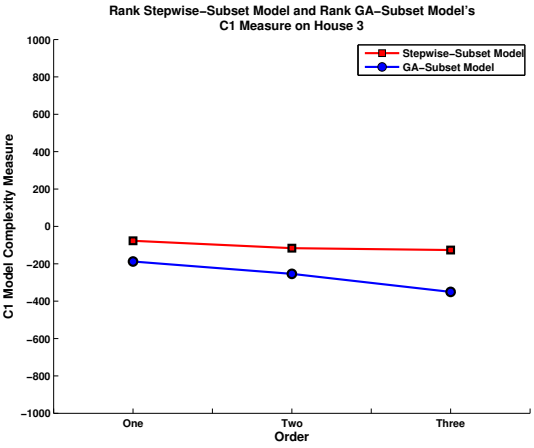
(a) Saturated Model's ICOMP



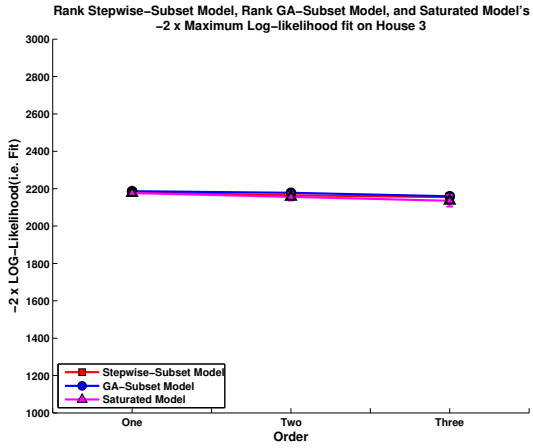
(b) GA and Stepwise Models' ICOMP



(c) Saturated Model's Complexity



(d) GA and Stepwise Models' Complexity



(e) Goodness-of-Fit

Figure 4.19: Experimental results for Campbell Creek House 3's Rank Models with dropped variables that having missing data.

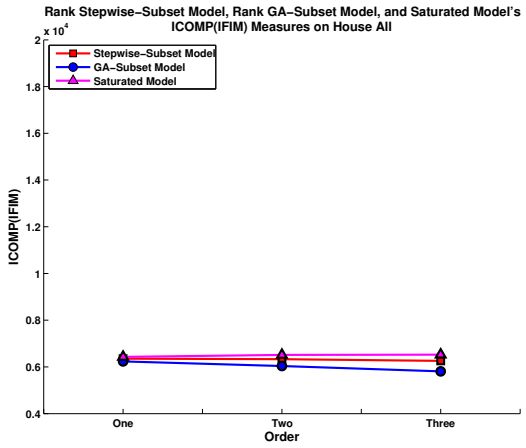
Table 4.1: Top 10 Sensors from the Voted Markov Order 1 models per house. The Markov Order 1 models were constructed by combining all the best Stepwise Selection subsets, using the voting process discussed in Section 4.5. Additionally, the best Stepwise Selection subsets were computed using datasets where variables with missing values were removed.

House 1	House 2	House 3	Across All
HW Tot	HW Tot	HP1 in Tot	HP1 in Tot
bathup lts Tot	bathup lts Tot	HP1 out Tot	HP1 out Tot
LVL1 lts Tot	LVL1 lts Tot	HP1 back Tot	HP1 in fan Tot
Kit tmp Avg	wash Tot	HP1 comp Tot	HP2 in Tot
BedB tmp Avg	LVL1 plg Tot	FanTech Tot	HP2 out Tot
Nrake1 tmp Avg	RoofN tmp Avg	solar HW pump Tot	FanTech Tot
Nrake2 tmp Avg	AtticN tmp Avg	bathup lts Tot	solar HW pump Tot
Srake1 tmp Avg	WallNcav tmp Avg	LVL1 lts Tot	HW Tot
Attic tmp Avg	BedM tmp Avg	bed Tot	bathup lts Tot
WashHot flow Tot	Bed2 tmp Avg	dryer Tot	LVL1 lts Tot

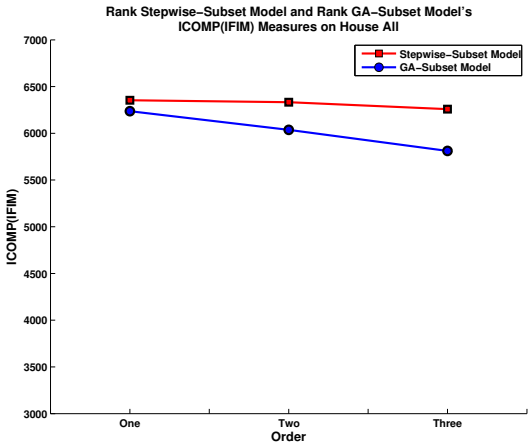
Lastly, the rank models created from the Genetic Algorithm and Stepwise Selection models across all houses perform the same as the results presented in Figure 4.9 and Figure 4.10. This shows that the ranking process is not degrading performance across all the houses, but it is not improving performance like it has for certain models on House 1 and House 2.

Figures 4.21, 4.22, 4.23, and 4.24 present the results from applying our sensor ranking technique to determine the best model, when missing values are set to zero. In addition, Tables 4.3 and 4.4 show the top ten sensors for both methods on Markov Order 1. All the Stepwise rank models shown in these figures are extremely similar to the previous Stepwise rank models (Figures 4.17, 4.18, 4.19, and 4.20) and do not provide performance increases, but do not decrease performance drastically either. In addition, the Genetic Rank Model on House 2 performs poorly in terms of model complexity, just like the Genetic Rank Model seen in Figure 4.18.

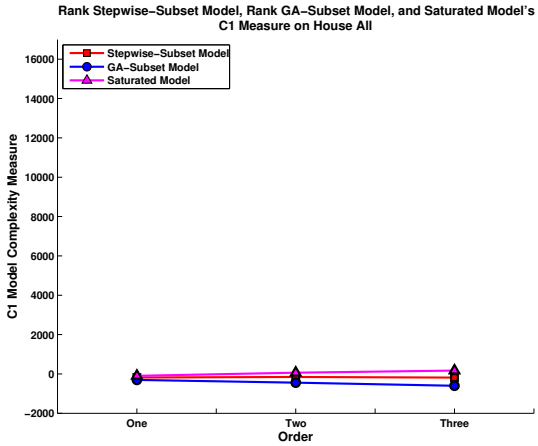
The key observation is that the Genetic Rank models in Figures 4.21, 4.23, and 4.24 present the best model for House 1, House 3, and across all houses. On House 1,



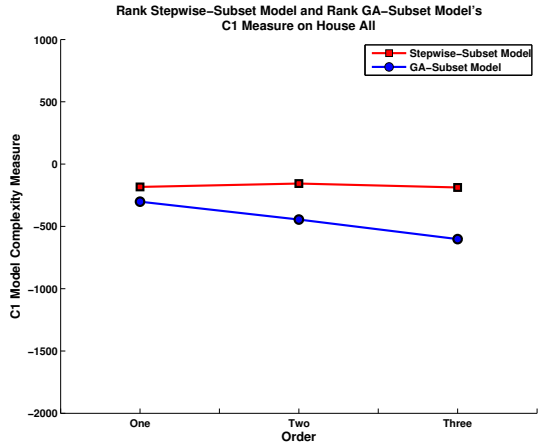
(a) Saturated Model's ICOMP



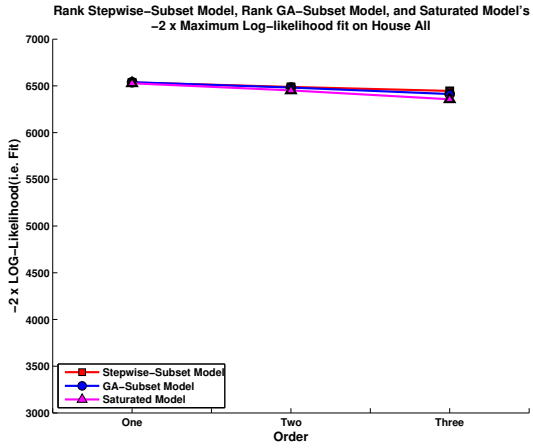
(b) GA and Stepwise Models' ICOMP



(c) Saturated Model's Complexity



(d) GA and Stepwise Models' Complexity



(e) Goodness-of-Fit

Figure 4.20: Experimental results for Rank Models across all houses with dropped variables that have missing data.

Table 4.2: Top 10 Sensors from the Voted Markov Order 1 models per house. The Markov Order 1 models were constructed by combining all the best Genetic Algorithm subsets, using the voting process introduced in Section 4.5. Additionally, the best Genetic Algorithm subsets were computed using datasets where variables with missing values were removed.

House 1	House 2	House 3	Across All
gar ext lts Tot	LVL1 lts Tot	bathup lts Tot	HP1 out Tot
LVL1 lts Tot	gar ext plg Tot	LVL1 lts Tot	FanTech Tot
bath plg Tot	CantFlr RH Avg	dryer Tot	LVL1 lts Tot
gar ext plg Tot	AtticN HFT Avg	wash Tot	bed Tot
bed Tot	HP1ret tmp Avg	micro Tot	dryer Tot
dish Tot	Attic RH Avg	range Tot	wash Tot
RoofS HFT Avg	FreshAir Flow Tot	FanTexh RH Avg	LVL1 plg Tot
fridge Tot	gar ext lts Tot	HW Tot	gar ext plg Tot
CondenHP1 Tot	CondenHP1 Tot	WallScav RH Avg	micro Tot
HP2sup RH Avg	WallScav RH Avg	FanTech ToT	dish Tot

the Markov Order 3 model provides the best goodness-of-fit compared to all previous results and has better model complexity than all previous models. The Genetic Rank model across all houses has the best goodness-of-fit and best complexity compared to all previous models as well. Lastly, the Genetic Rank Model improves model complexity greatly on House 3, making it the best performing model in terms of ICOMP(IFIM) for all Markov Orders.

4.6.6 Ground Truth Comparison

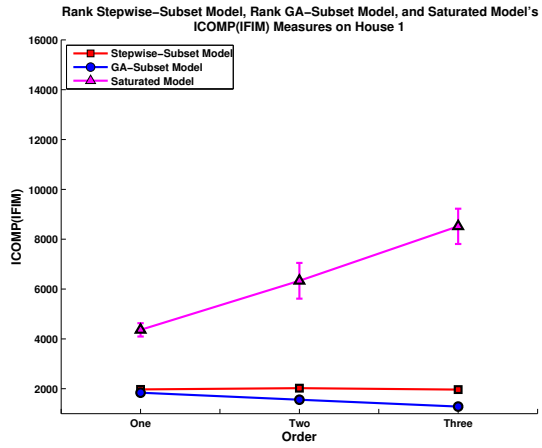
An advantage of our model selection approach is that it can allow a practical search over a large solution space to find good solutions that work well in practice. Comparing it to the "Ground Truth" solution is computationally infeasible. Nevertheless, it is informative to calculate the exact solution for small problems, in order to provide comparative results to our approach. We, therefore, calculated the best sensor subsets, "Restricted Ground Truth," with cardinality up to four. We refer to these sensor subsets as "Restrict Ground" because they are globally optimal solutions

Table 4.3: Top 10 Sensors from the Voted Markov Order 1 models per house. The Markov Order 1 models were constructed by combining all the best Stepwise Selection subsets, using the voting process discussed in Section 4.5. Additionally, the best Stepwise Selection subsets were computed with missing data values set to zero.

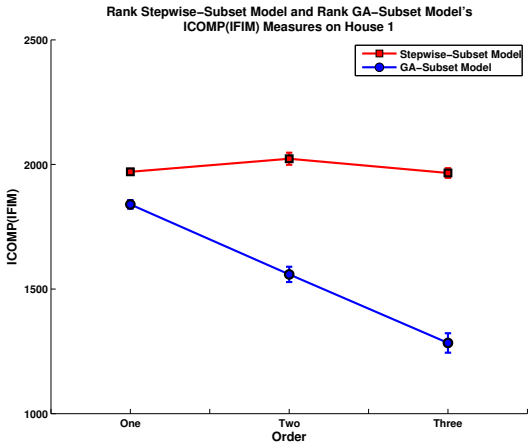
House 1	House 2	House 3	Across All
bathup lts Tot	bathup lts Tot	HP1 in Tot	bathup lts Tot
bed Tot	LVL1 lts Tot	HP1 out Tot	LVL1 lts Tot
dish Tot	wash Tot	HP1 back Tot	wash Tot
range Tot	LVL1 plg Tot	HP1 comp Tot	micro Tot
WallScav tmp Avg	RoofN tmp Avg	bathup lts Tot	dish Tot
BedM tmp Avg	AtticN tmp Avg	LVL1 lts Tot	CantFlr tmp Avg
Bed3 tmp Avg	WallNcav tmp Avg	bed Tot	BedM tmp Avg
BedB tmp Avg	BedM tmp Avg	wash Tot	Bed3 tmp Avg
Nrake1 tmp Avg	Bed2 tmp Avg	micro Tot	Bed2 tmp Avg
Nrake2 tmp Avg	Mbath tmp Avg	RoofS tmp Avg	BedB tmp Avg

Table 4.4: Top 10 Sensors from the Voted Markov Order 1 models per house. The Markov Order 1 models were constructed by combining all the best Genetic Algorithm subsets, using the voting process introduced in Section 4.5. Additionally, the best Genetic Algorithm subsets were computed with missing data values set to zero.

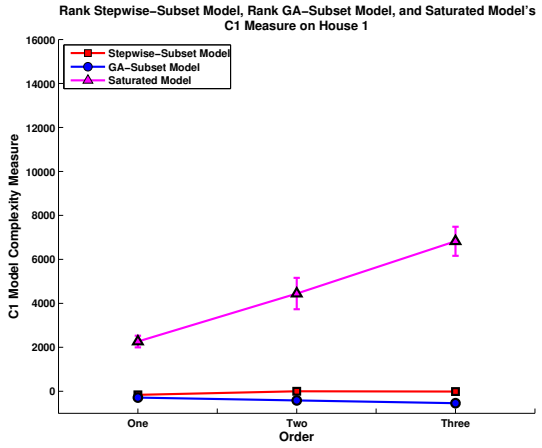
House 1	House 2	House 3	Across All
HWcold tmp Avg	wash Tot	wash Tot	HWhot tmp Avg
dish Tot	HWcold tmp Avg	HP1 out Tot	washHot tmp Avg
LVL1 lts Tot	gar ext lts Tot	WashHot flow Tot	HP1ret RH Avg
HWhot tmp Avg	gar ext plg Tot	SlrW1 Avg	Nrake5 tmp Avg
TrueNetEnergy	bed Tot	gar ext lts Tot	dishHot tmp Avg
BedB tmp Avg	CantFlr RH Avg	HP1 comp Tot	WallScav RH Avg
HP2sup tmp Avg	fridge Tot	HWHXtoTank tmp Avg	LVL1 lts Tot
RoofS HFT Avg	Nrake5 tmp Avg	AtticFlrS HFT Avg	HWcold tmp Avg
bathup lts Tot	Shower tmp Avg	dryer Tot	CondenHWHP Tot
gar ext lts Tot	WallScav RH Avg	bed Tot	HP1 in fan Tot



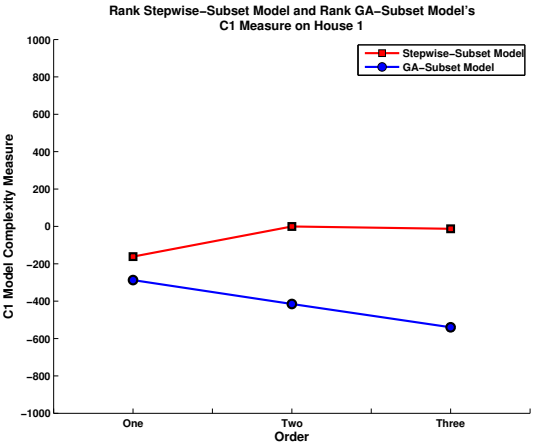
(a) Saturated Model's ICOMP



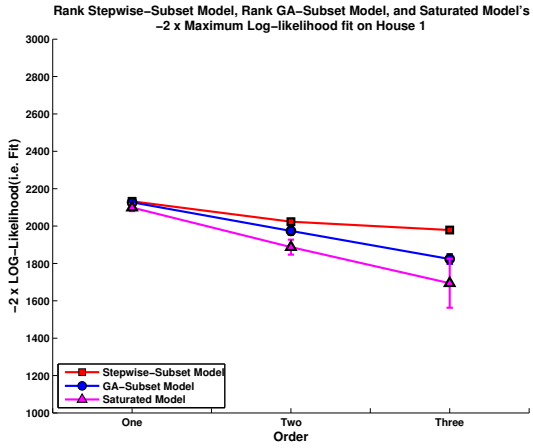
(b) GA and Stepwise Models' ICOMP



(c) Saturated Model's Complexity

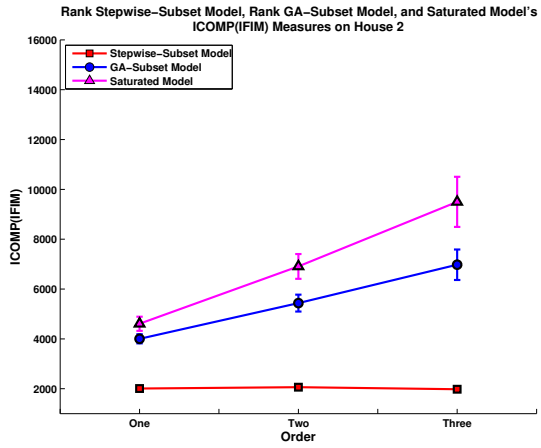


(d) GA and Stepwise Models' Complexity

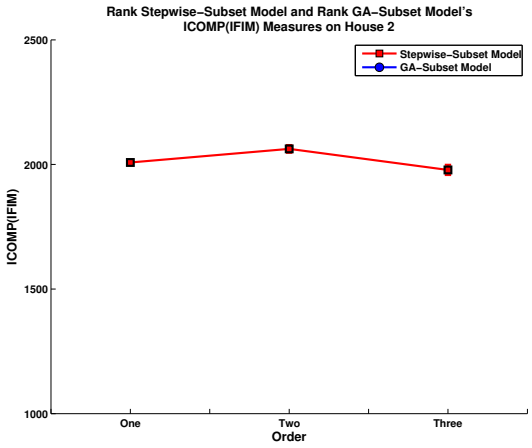


(e) Goodness-of-Fit

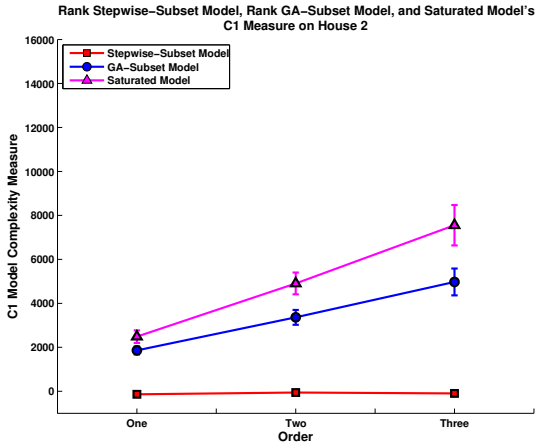
Figure 4.21: Experimental results for Campbell Creek House 1's Rank Models with missing data values set to zero.



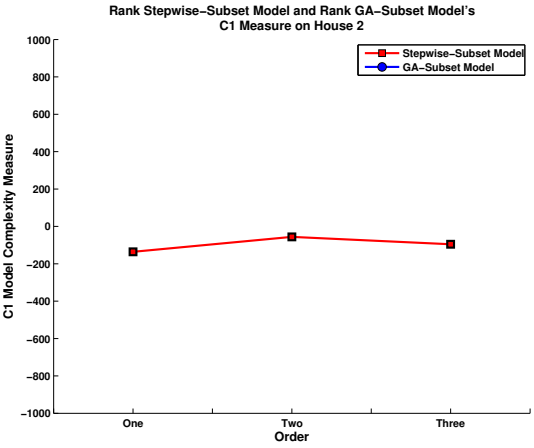
(a) Saturated and GA Models' ICOMP



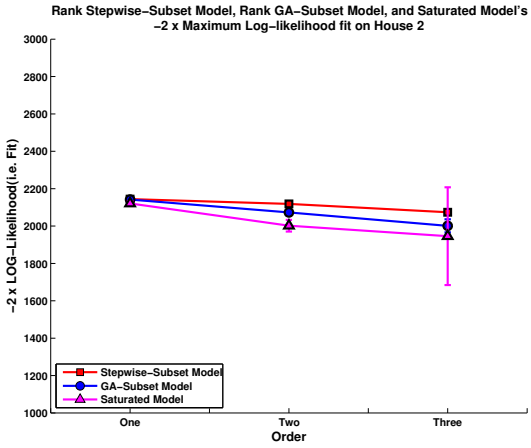
(b) Stepwise Model's ICOMP



(c) Saturated and GA Models' Complexity

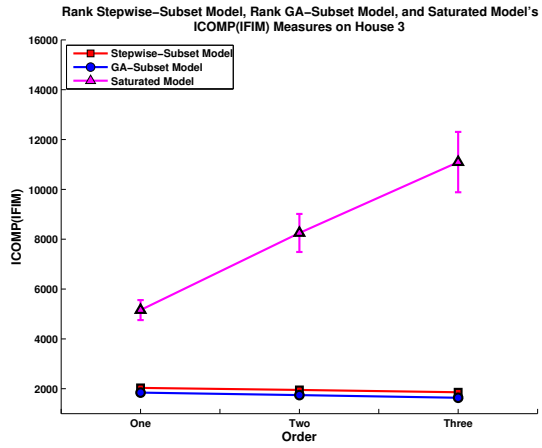


(d) Stepwise Model's Complexity

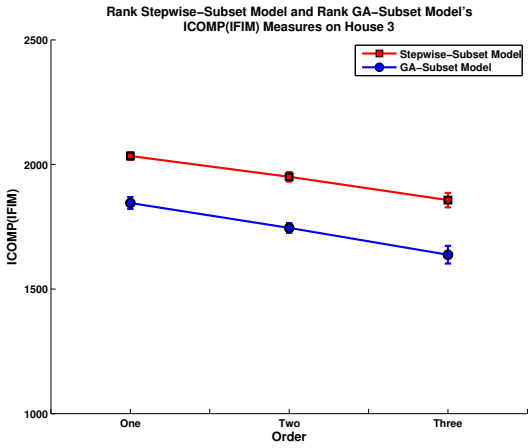


(e) Goodness-of-Fit

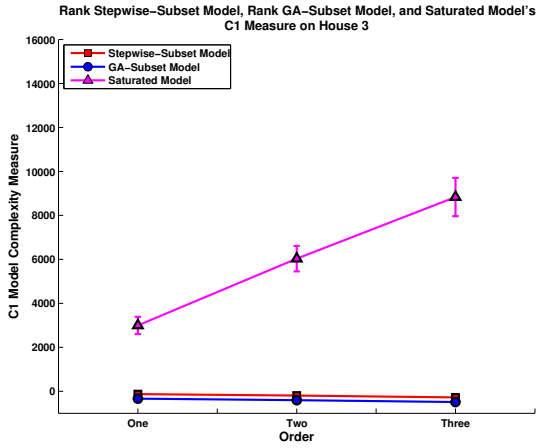
Figure 4.22: Experimental results for Campbell Creek House 2's Rank Models with missing data values set to zero.



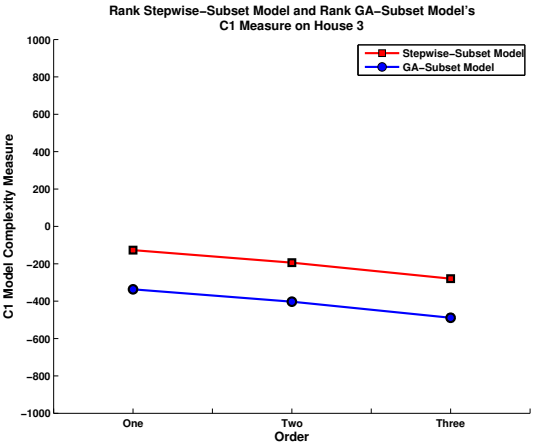
(a) Saturated Model's ICOMP



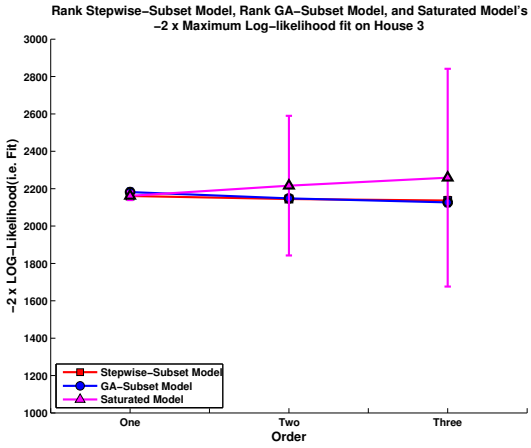
(b) GA and Stepwise Models' ICOMP



(c) Saturated Model's Complexity

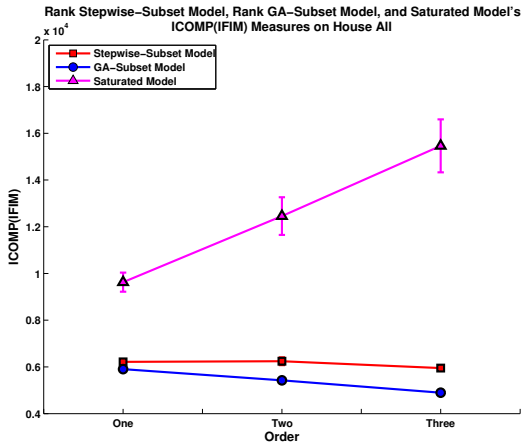


(d) GA and Stepwise Models' Complexity

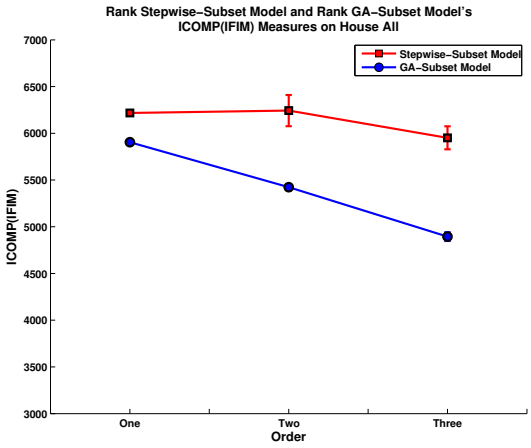


(e) Goodness-of-Fit

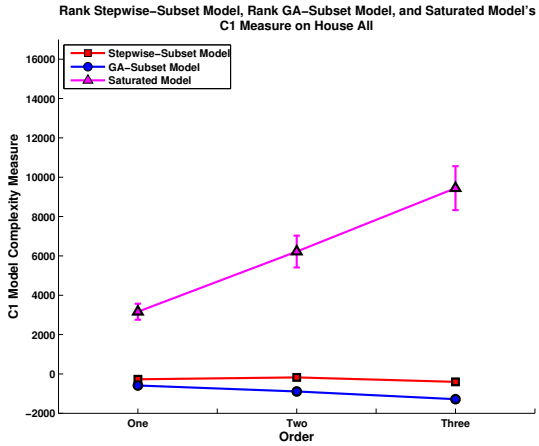
Figure 4.23: Experimental results for Campbell Creek House 3's Rank Models with missing data values set to zero.



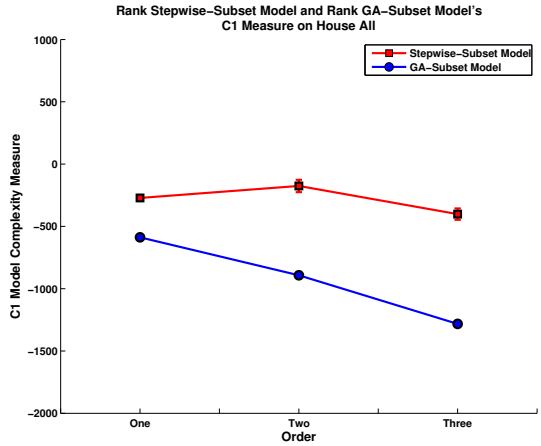
(a) Saturated Model's ICOMP



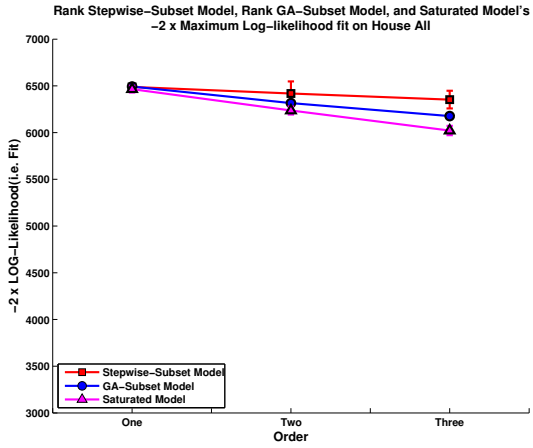
(b) GA and Stepwise Models' ICOMP



(c) Saturated Model's Complexity



(d) GA and Stepwise Models' Complexity



(e) Goodness-of-Fit

Figure 4.24: Experimental results for applying Rank Models across all houses with missing data values set to zero.

Table 4.5: The House 1 Table compares House 1's "Restricted Ground Truth" subsets against the best Markov Order 1 Rank Model seen in Figure 4.17 and against the Top 10 Sensor list for House 1 in Table 4.2. The House 2 Table compares House 2's "Restricted Ground Truth" subsets against the best Markov Order 1 Genetic Algorithm Model seen in Figure 4.6 and the best Top 10 Sensor list for House 2 (Table 4.1). Variables with missing data were removed for these comparisons.

Houses 1

Best Four Sensors		Best Four	Best Model	Top 10 Sensors
HW Tot	RMSE	1162.98±28.8959	1053.13±25.6179	1604.09±44.8236
LVL1 LTS Tota	MAPE	41.098±0.974165	35.9376±1.15145	67.0153±1.89464
Dryer Tot	CV	41.4157±0.661293	37.505±0.642408	57.1235±1.09945
EstimateNetEnergy	MBE	0.0393377±1.06234	0.0656687±1.09142	0.668442±1.76908
	ICOMP(IFIM)	2166.43±1.63914	1903.04±11.9395	2139.39±3.54907

Houses 2

Best Four Sensors		Best Four	Best Model	Top 10 Sensors
HP1 back Tot	RMSE	799.769±18.2186	687.117±11.7289	972.158±20.0695
HP1 comp Tot	MAPE	38.1217±1.45629	32.0021±1.27032	54.0946±2.11265
LVL1 Lts Tot	CV	43.0711±0.81126	37.0045±0.433794	52.3549±0.836793
Wash Tot	MBE	0.207401±0.642866	-0.0373696±0.630909	0.0936422±1.06155
	ICOMP(IFIM)	2165.98±0.896068	1881.83±7.9845	2151.16±2.36627

Table 4.6: The House 3 Table compares House 1's "Restricted Ground Truth" subsets against the best Markov Order 1 Rank Model seen in Figure 4.19 and against the Top 10 Sensor list for House 3 in Table 4.2. The Across All Table compares the "Restricted Ground Truth" subsets across all houses against the best Markov Order 1 Rank Model seen in Figure 4.20 and the best Top 10 Sensor list across all houses (Table 4.2). Variables with missing data were removed for these comparisons.

Houses 3

Best Four Sensors		Best Four	Best Model	Top 10 Sensors
HP1 in Fan Tot	RMSE	761.539±14.8683	660.814±9.79829	839.712±21.7697
Wash Tot	MAPE	42.7805±1.02704	34.5621±0.84633	51.3902±0.751316
CantFlr tmp Avg	CV	48.7213±0.716605	42.2796±0.56895	53.7194±1.01782
Kit tmp Avg	MBE	-0.173105±0.641638	-0.114178±0.949378	-0.172517±1.34433
	ICOMP(IFIM)	2158.29±0.672128	1988.92±5.9571	2121.49±3.46866

Houses All

Best Four Sensors		Best Four	Best Model	Top 10 Sensors
HP1 in Tot	RMSE	988.897±12.9624	886.097±14.0127	1196.59±12.0652
HP1 out Tot	MAPE	44.8416±0.572332	36.8267±0.6752	52.505±0.979291
HP2 in Fan	CV	47.6745±0.683444	42.7179±0.676298	57.6865±0.569256
LVL1 lts Tot	MBE	0.416336±0.500918	0.2335±0.526657	0.653978±0.819731
	ICOMP(IFIM)	6531.88±1.0026	6234.23±6.21427	6502.32±8.61767

Table 4.7: The House 1 Table compares House 1's "Restricted Ground Truth" subsets against the best Markov Order 1 Rank Model seen in Figure 4.21 and against the best Top 10 Sensor list for House 1 (Table 4.4). The House 2 Table compares House 2's "Restricted Ground Truth" subsets against the best Markov Order 1 Genetic Algorithm Model seen in Figure 4.8 and the best Top 10 Sensor list for House 2 (Table 4.3). Missing data values were set to zero for these comparisons.

Houses 1

Best For Sensors		Best Four	Best Model	Top 10 Sensors
HW Tot	RMSE	1128.13±33.8483	930.84±36.4096	1137.83±35.4406
Dryer Tot	MAPE	41.1941±0.626537	30.8483±0.808462	40.4656±0.835134
WashHot tmp Avg	CV	39.8526±0.962945	32.8796±1.02213	40.1942±0.988229
EstimateNetEnergy	MBE	0.251124±1.39711	0.205174±1.01119	0.0539458±1.25084
	ICOMP(IFIM)	2167.74±3.79636	1842.51±23.959	2126.39±3.46705

Houses 2

Best Four Sensors		Best Four	Best Model	Top 10 Sensors
Nrake2 tmp Avg	RMSE	811.081±17.7065	696.098±11.3641	928.39±17.1124
Nrake4 tmp Avg	MAPE	38.6717±1.56478	32.089±1.64436	49.495±1.8304
BonusFlr HFT Avg	CV	43.6259±0.944585	37.4407±0.571278	49.9339±0.802609
EstimateNetEnergy	MBE	0.476007±1.0076	0.133703±0.910032	-0.12749±0.905776
	ICOMP(IFIM)	2183.98±1.17292	1823.16±12.9824	2157.26±2.38136

Table 4.8: The House 3 Table compares House 1's "Restricted Ground Truth" subsets against the best Markov Order 1 Rank Model seen in Figure 4.23 and against the best Top 10 Sensor list for House 3 (Table 4.4). The Across All Table compares the "Restricted Ground Truth" subsets across all houses against the best Markov Order 1 Rank Model seen in Figure 4.24 and the best Top 10 Sensor list across all houses (Table 4.4). Missing data values were set to zero for these comparisons. SSE stands for Sum of Squared Error.

Houses 3

Best Four Sensors		Best Four	Best Model	Top 10 Sensors
Wash Tot	RMSE	738.163±29.3591	682.365±61.5981	780.885±28.3228
FanTsup tmp Avg	MAPE	41.7311±0.882484	35.0396±0.951685	45.8574±0.867198
HWcolToHX tmp Avg	CV	47.4257±1.57935	43.8143±3.4824	50.1698±1.43239
EstimateNetEnergy	MBE	0.250537±1.10433	0.582986±0.932043	0.612654±0.959586
	ICOMP(IFIM)	2156.82±1.82757	1864±38.3738	2125.13±1.71237

Houses All

Best Four Sensors		Best Four	Best Model	Top 10 Sensors
HW Tot	RMSE	957.374±6.45585	828.045±6.0353	1233.62±16.5752
Dryer Tot	MAPE	42.282±1.1681	35.747±0.860365	58.1792±0.92238
AtticN RH Avg	CV	45.9085±0.490904	39.706±0.346226	59.1546±0.884414
EstimateNetEnergy	MBE	-0.181538±0.669515	-0.121514±0.537264	-0.179211±0.63554
	ICOMP(IFIM)	6532.19±1.50444	5907.77±15.648	6496.73±1.14173

to a smaller problem. Tables 4.5 and 4.6 show the "Restricted Ground Truth" subsets for datasets with removed missing data variables, while Tables 4.7 and 4.8 show the "Restricted Ground Truth" subsets for datasets with missing data values set to zero. These subsets were computed in a brute force fashion by selecting the best subset from all possible subsets that minimized the linear regression's residual SSE (Sum Squared Error). These tables compare the "Restricted Ground Truth's" energy prediction metrics and ICOMP scores against the best found Genetic Algorithm Models with Markov Order 1 and the best Top 10 Sensor lists on each respective dataset. This means that we are only comparing "Restricted Ground Truth" results for a dataset against models that were found using the same dataset. The "Top 10 Sensors" results in Tables 4.5 and 4.6 were generated using the sensor listings found in Table 4.2 for House 1, 3, and across all, while the results for House 2 were generated using the sensor listings found in Table 4.1. The "Top 10 Sensors" results in Tables 4.7 and 4.8 were generated using the sensor listings found in Table 4.4.

Analyzing Tables 4.5 and 4.6 shows that there is a large difference in performance between the "Restricted Ground Truth" and the best Genetic Algorithm models with Markov Order 1, seen in Figures 4.17, 4.6, 4.19, and 4.20. The Genetic Algorithm Models have much better performance in terms of energy prediction metrics and ICOMP, which implies that the best performing optimal subset is larger than the ones we have computed. However, these best performing approximations use 50 or more sensors. This makes it very difficult to estimate the best performing optimal subset's actual size and to estimate whether one can feasibly compute it directly.

Comparing the same "Restricted Ground Truth" results with the best "Top 10 Sensor" lists results shows that the voting scheme is able to produce lower ICOMP values, but overall worse energy prediction results. This implies solving for a small subset directly is better than selecting a small subset using our variable ranking procedure. However, if one is concerned about the best subset being generalizable, then one can solve directly for the best subset using ICOMP as the criteria function rather than SSE.

Tables 4.7 and 4.8 illustrate that the best Genetic Algorithm Models with Markov Order 1 in Figures 4.21, 4.8, 4.23, and 4.24 have better energy prediction metrics and better ICOMP scores than the "Restrict Ground Truth" subsets. This provides additional evidence that the best performing optimal subset is larger than four sensors. However, comparing the "Top 10 Sensor" results with the same "Restricted Ground Truth" results further reinforces that solving for a small subset directly is better than using our ranking procedure to select a small set of variables.

In summary, if one wishes to find the best performing optimal subset, it is computationally infeasible in the general case because computing sensor subsets with four sensors takes three hours, five sensors takes three to four days, six sensors takes 75 days, and seven or more sensors takes years. However, one can produce reasonably good approximations using our approach. On the other hand, if one is interested in solving for a small optimal subset and one has enough computing resources, then it is best to compute it directly.

4.7 Results Summary

The results presented in Sections 4.6.1, 4.6.2, 4.6.3, and 4.6.4 show that the Genetic Algorithm with the ICOMP(IFIM) model criteria as the fitness function is able to find better models than the Stepwise Selection method. In addition, these sections show that the best models were found with Markov Order 3, and that setting missing values equal to zero is better than removing sensors that have missing values. Applying our voting technique to the Genetic Algorithm models allows us to find a better model (Figures 4.21, 4.23, and 4.24) than the best single Genetic Algorithm model (Figures 4.4, 4.12, and 4.16) on House 1, House 3, and across all houses. Therefore, on future homes we recommend comparing the best single Genetic Algorithm model with the model made from our voting process and then selecting the best performing model from these two. However, if one is interested in finding the best model for a sufficiently small sensor subset, e.g., up to 5 sensors, it is recommended that one solve for this

best model directly, because it should be computationally feasible to test all possible subsets.

4.8 Computer Science Contribution Summary

While the wrapper methods and our novel feature selection process were used to solve a domain specific problem, these methods are all general and application independent, which is well demonstrated in the literature Bozdogan and Haughton (1998); Bozdogan (2003). This means our voting method combined with ICOMP will out perform the traditional stepwise feature selection method on most problems[¶]. In addition, our voting method usually out performs stand alone ICOMP feature selection. This result imply that analyzing the model selection criteria's variance and stability has a large impact on the overall selection process, which is directly incorporated into our voting algorithm.

[¶] This statement is based on ICOMP out performing stepwise, backwards, and forwards selection in the literature and the results in this dissertation

Chapter 5

Simulation Approximation

Even with a 40% reduction in runtime from E+ 6.0 to E+ 7.0 (Chapter 2), manually tuning E+ building models is still a very slow and tedious process. For example, an engineer manually tuning a simulation is not likely to wait the 2-3 minutes required to run an E+ simulation before proceeding to the next tuning step; likewise, the Autotune methodology runs 1024 simulations, which at only three minutes per simulation would require over two days. The overall computational burden is generally reduced by constructing surrogates or meta-models Ong et al. (2003); Jin et al. (2001); Qian et al. (2006). Surrogates are simplified models that approximate the original model and require less computational resources. These surrogates are generally statistically generated models, but there are other methods for producing simplified approximation models, such as coarse-grid approximations Kaminski et al. (1999) or proper orthogonal decomposition Willcox and Peraire (2002). In this work, we focus solely on statistical and machine learning generated surrogates.

While a surrogate model provides computational advantages, its calibration utility is completely dependent upon the model's accuracy. However, to the best of our knowledge, very little work focuses on whole-building E+ simulation surrogates within the building spaces domain. In fact, the few available studies only explored a limited number of envelope parameters, operation parameters, and outputs. Tresidder et al.

(2011, 2012) used 10 envelope parameters, while Tian and Choudhary (2012) used 16 envelope and operational parameters. All three studies' surrogate models only estimate a single output. A vast majority of surrogate studies in other engineering disciplines only use a limited number of inputs and outputs as well. Therefore, it is difficult to ascertain how well a surrogate model can approximate E+ on a large-scale relative to other studies.

This study explores machine learning and statistical techniques for constructing large scale E+ surrogate models. Two surrogate models were constructed using Feed Forward Neural Networks (FFNN) and Lasso regression. The surrogate models were generated using three very large E+ simulation data sets for a residential building. The data sets cover fine grain parameter sweeps over seven envelope parameters with 80 simulation outputs and coarse broad parameter sampling over 156 envelope parameters with 90 simulation outputs. The data sets are covered in more detail in Section 2.2.3. Using these data sets, we evaluate the two surrogate models' abilities to approximate larger E+ simulations in comparison to previous studies.

The remainder of this chapter is organized as follows: Section 5.1 discusses our approximation methods; Section 5.2 presents our evaluation criteria; Section 5.3 presents our approximation results; Section 5.4 discusses our prediction results and interesting observed phenomena found through the experimentation process; and Section 5.5 summarizes our findings and conclusions.

5.1 Approach

We explored two different methods for approximating E+. The first method utilizes standard FFNN with a modified training process. We adjusted the training process to accommodate our large data sets, which ultimately allows computationally tractable large scale FFNN learning. FFNN background information is presented in Section 3.4 and our training procedure is presented in Section 5.1.1.

The second approach uses Lasso regression, a linear model, which has the ability to automatically select relevant input variables. This allows us to determine whether we have sufficient information within our datasets to produce predictions and determine if a complex nonlinear model, i.e. FFNN, is actually required. However, the standard Lasso regression learning algorithms are not designed to support large-scale learning. To overcome this difficulty, we use a recently developed decentralized optimization framework, called Alternating Direction Method of Multipliers (ADMM) Boyd et al. (2011). This method supports arbitrary large-scale learning by dividing the original problem into smaller local optimization problems. These problems are either distributed across multiple computers or solved locally on a single memory constrained computer that uses the hard drive as additional storage. We discuss general Lasso regression information in Section 5.1.2, ADMM's framework in Section 5.1.3, and Lasso regression's ADMM formulation in Section 5.1.4. Finally, we discuss how we select the best parameter settings for the Lasso and FFNN models (Section 5.1.5).

5.1.1 Large Scale-Feed Forward Neural Network Training

There are two gradient based methods for training FFNN: stochastic and batch. The stochastic method uses a single observation to compute the gradient and update the network. This method is extremely scalable to large data sets, because it makes updates per training example. However, stochastic gradient descent only approximates the gradient using local information, which means it lacks the global information required to follow the objective function's true gradient. While this allows the stochastic gradient descent method to scale well, it may produce less accurate models, because an approximate gradient is substituted for the exact gradient.

The batch gradient descent method uses all training examples to compute the gradient and update the network. This method is much less scalable than the stochastic method, because it has to process all the examples per update. Computing

the gradient using the entire data set allows this method to produce better gradient estimates, which may lead to more accurate networks. However, this method is not typically practical since it will require hundreds of passes over a gigabyte data set.

Given that both approaches provide different useful benefits, we used a hybrid method for training the FFNN. The hybrid method we used is a stochastic gradient descent that performs updates using mini-batches, rather than updates per single training example. This allows us to balance training time performance and gradient estimation quality. In our approach, we select a random simulation and divide the simulation into mini-batches. Before constructing the mini-batches, each sampled simulation is randomly shuffled. Randomly sampling the simulations and shuffling the internal simulation data provides the stochastic gradient characteristics. In addition, making network updates per randomized mini-batch provides a pseudo batch gradient descent characteristic. In summary, we sample a simulation, randomize the simulation's data vectors, divide the data into mini-batches, and update the network using each mini-batch.

5.1.2 Lasso Regression

Standard Lasso regression fits a linear model by modifying a multiplicative weighting factor for each input and adding the weighted inputs to create the outputs. The final model has the same functional form as linear regression and ridge regression, but the learning criteria inserts a term to penalize the absolute size of the regression coefficients. This allows automatic feature selecting and also overcomes standard regression problems with over weighting highly correlated predictors. The following equation shows the Lasso regression optimization criteria:

$$\sum_{i=1}^{n}(y_i - b - w^{\mathsf{T}}(\vec{x_i})) + \lambda||w||$$

where $\vec{x_i}$ is an input vector, y_i is the response, w is the model weights, b is the y intercept, and λ produces a trade-off between fitting and sparsity. Increasing λ leads to a model with more zero value weights. This means, under an appropriate λ value, irrelevant inputs in $\vec{x_i}$ are ignored, which results in a sparser and more robust model. Note that robustness is defined based on the idea that a simplistic model is most likely to generalize to new scenarios, which is based on model complexity studies Akaike (1973); Schwarz (1978); Bozdogan and Haughton (1998).

Lasso regression can easily be extended to nonlinear functions using kernels, but we have not explored that direction within this work. There are two reasons to use the linear model over the nonlinear model in this particular study. First, we have observed excellent performance using nonlinear FFNNs, but the computational training time for these models is fairly high. Depending on the optimization method, the Lasso regression's linear model can be solved much faster. In addition, the linear model will indicate whether a nonlinear model is required. If a linear model is sufficient, then we can substantially reduce the overall training time for larger data sets.

The second reason is based on Lasso's regression variable selection capabilities. This particular feature allows the learned models to be directly interpretable based on the learned values for w. More importantly, it allows us to analyze which information is currently important for making predictions, and can help us ascertain if required information is missing within the data set.

5.1.3 Alternating Direction Method of Multipliers

Lasso regression models are learned using quadratic programming methods, which include interior point (Nesterov et al., 1994), gradient methods, and many more. However, traditional quadratic programming methods scale poorly to large datasets (Joachims, 1999b). Improving optimization algorithms' performance and scalability is an active research area. As a result, there are many different methods that are able to scale well to different problem types (Chang and Lin, 2011b; Collobert

and Bengio, 2001; Teo et al., 2007; Franc and Sonnenburg, 2009). For example, the Convex Bundle Cutting Plane (CBCP) method (Teo et al., 2007) is a highly scalable optimization algorithm that uses piecewise linear components to iteratively approximate the criteria function and find a solution.

The methods investigated vary in their scalability in relation to parallelizing across multiple processing cores and utilizing the underlying hardware efficiently. For example, the CBCP method parallelizes fairly well by splitting large data sets across multiple computers, but the parallel algorithm uses a master-slave paradigm, in which the slave computers solve subproblems, while the master computer aggregates the sub-solutions and solves an additional optimization problem over the available information. While the master computer is solving its optimization problem, the slave computers are idle, which reduces overall resource efficiency. In order to maximize resource utilization, we elected to use Alternating Direction Method of Multipliers (ADMM) (Boyd et al., 2011) over other equally capable distributed optimization methods because it does not use a master-slave paradigm. While the following detailed ADMM description illustrates solving a redundant secondary optimization problem per computer, the optimization problem in practice is extremely light-weight, which makes it more efficient to redundantly solve the problem locally, rather than communicate the solution to slave computers.

ADMM is a fully decentralized distributed optimization method that can scale to very large machine learning problems. ADMM solves the optimization problem directly without using approximations during any phase of the optimization process. The optimization process works by splitting the criteria function into separate sub-problems and optimizing over those individual problems with limited communication. The following is a formal explanation from (Boyd et al., 2011):

$$\text{minimize} f(x) + g(z)$$

with the constraints $Ax+Bz = c$, where c is a targeted response or agreed target value that correlates the two functions. In addition, f and g are convex, closed, and proper functions. The functions $f(x)$ and $g(z)$ are independent, which means both can be minimized in parallel. In addition, the equality constraint provides consensus across the two functions. More importantly, (Boyd et al., 2011) introduces the following Lagrangian for this particular optimization problem:

$$L_p(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + (\rho/2)||Ax + Bz - c||_2^2$$

where ρ defines a tunable parameter that determines the trade off between violating the equality constraint and fitting the target function. Boyd et al. (Boyd et al., 2011) recommends setting ρ to 1 and notes that finding ideal ρ values is based on manual trial and error, rather than an automated selection process. After some additional algebraic simplifications of the above Lagrangian, the final ADMM optimization process is as follows:

$$x^{k+1} = \underset{x}{\text{argmin}}(f(x) + \frac{\rho}{2}||Ax + Bz^k - c + u^k||_2^2) \tag{5.1}$$

$$z^{k+1} = \underset{z}{\text{argmin}}(g(z) + \frac{\rho}{2}||Ax^{k+1} + Bz^k - c + u^k||_2^2) \tag{5.2}$$

$$u^{k+1} = u^k + x^{k+1} - z^{k+1} \tag{5.3}$$

where superscript k indicates the current iteration values and $k + 1$ represents the values for the next iteration. Iterating over these optimization equations provides guaranteed convergence, and establishes a method to minimize x and z independently with limited communication between the two optimization problems.

The above form can be deconstructed further into multiple sub-problems across $f(x)$ by sub-dividing the function across the independent components within x. This creates independent sub-problems that are solved locally via the first minimization step (Eq. 5.1), which allows multiple computers to optimize $f(x)$ locally, and pass information to other computers or processes about x^{k+1}, resulting in a global

optimization over z^{k+1} at each individual process (Eq. 5.2). This means all processes can work to optimize and compute their individual updates by only communicating their local beliefs for x^{k+1}. Eq. 5.3 represents an updated cost penalty trade off versus accuracy per data element, which is a standard component with all penalized optimization problems.

5.1.4 Large-Scale Lasso Regression

There exist several common substructures for constrained convex optimization problems (Boyd et al., 2011). In particular, the general minimization problem is defined as follows:

$$\text{minimize} f(x)$$

with the constraints $x \in C$, where C defines a constrained solution space. This general minimization problem is formulated as the following under ADMM:

$$\text{minimize} f(x) + g(z)$$

with the constraint $x - z = 0$, where g is an indicator function. Using an indicator function for g allows ADMM to represent the original convex optimization constraints, and the $x - z = 0$ constraint guarantees that the x that minimizes $f(x)$ obeys the original constraints.

While (Boyd et al., 2011) used this general solution format to solve many different convex optimization problems, we are only interested in the version used to solve Lasso regression. The distributed optimization steps for solving large scale linear

Lasso regression problems are presented below*:

$$x_i^{k+1} = \underset{x_i}{\text{argmin}}(\frac{1}{2}||A_i x_i - b_i||_2^2 + \frac{\rho}{2}||x_i - z^k + u_i^k||_2^2) \tag{5.4}$$

$$z^{k+1} = S_{\frac{\lambda}{\rho N}}(\bar{x}^{k+1} + \bar{u}^k) \tag{5.5}$$

$$u_i^{k+1} = u_i^k + x_i^{k+1} - z^{k+1} \tag{5.6}$$

The individual local subproblems are solved using ridge regression (Eq 5.4), and the global z (Eq. 5.5) values are computed by evaluating a soft thresholding function S (Eq. 5.7).

$$S_{\frac{\lambda}{\rho N}}(v) = max(0, v - \frac{\lambda}{\rho N}) - max(0, -v - \frac{\lambda}{\rho N}) \tag{5.7}$$

N represents the total number of training examples and λ is the Lasso regularization parameter. The soft thresholding function (Eq. 5.7) applies the Lasso regression sparsity constraints over z, which are incorporated into the local subproblem solutions on the next optimization iteration.

The key advantage behind this particular Lasso regression formulation is that the main heavy lifting step is solved exactly once. The direct method for computing x_i^{k+1} requires computing $(A^\intercal A + \rho I)^{-1}$. The resulting matrix never changes throughout the entire optimization process. Storing this result allows the distributed optimization method to perform a very computationally intensive task once and reduce all future x_i^{k+1} computations steps. Caching the values used to compute x_i^{k+1} to disk allows a 2.2GHz Intel Core i7 laptop to solve a univariate 3.9GB Lasso regression problem in approximately 17 minutes. The best FFNN model with 15 hidden units and 10 outputs completed training in 24 hours on the same 3.9GB data set.

* This version assumes we are only splitting the optimization problem across the training samples, and not the features. It is possible to split across both. (Boyd et al., 2011) presents the ADMM formulation for supporting this functionality.

5.1.5 Model selection

The final Lasso regression model's performance is greatly dependent upon the λ value used during the training process. Similarly, a FFNN's performance is greatly dependent upon the total number of hidden units selected. Selecting a λ value that is too small can result in overfitting, while selecting a value that is too large can lead to underfitting. The same possibilities apply to FFNN, but selecting too few hidden units can lead to underfitting, and selecting too many can lead to overfitting.

Selecting the best parameter setting is achieved by evaluating a model selection criteria, which measures how well the learned model will generalize to unseen examples. There are several different model selection techniques. For example, cross-validation methods estimate how well a model generalizes to unseen data by periodically testing the current model on a validation set. An online validation process is one that terminates the learning process when the model begins to perform poorly on the validation set. K-Folds cross-validation is another approach that divides the data into K partitions, and builds a model using $K-1$ partitions as training data. The omitted partition is used to evaluate the model as testing data. This training and testing process is repeated such that each partition is used as the testing set at least once. K-Folds' primary advantage over other methods is that it can provide an almost unbiased error estimate for any particular model as K approaches the data set's sample size Cawley and Talbot (2010).

Ideally, we would use a combination of cross-validation and K-Folds to select the best parameter values. Cross-validation enables online FFNN learning termination, and K-Folds facilitates selecting the correct number of hidden units. On the other hand, Lasso regression uses the validation set to select λ and K-Folds to approximate the model's overall error. However, the large data set makes K-Folds cross-validation computationally difficult to perform. Therefore, we elected to use a pure cross-validation method for parameter selection. Each model has a training set, a single validation set, and a testing set. For the FFNN models, we use the validation set

to prevent overfitting and we compare hidden unit settings using the unseen testing data. The Lasso regression models use the validation set to select the best λ value by picking the one that maximizes prediction accuracy for the validation set. This parameter selection method allows us to estimate the Lasso regression model's true prediction capabilities by using the unseen testing data. In addition, the testing data results can be directly compared with the FFNN results.

5.2 Methods

5.2.1 Experimental Design

Given the need for scalability and performance, we optimized the FFNN network structure by determining that the maximum number of outputs per network should be 10. This means that we use eight FFNNs to model the FG data set's 80 outputs, and nine FFNNs to model the MO1 data set's 90 outputs[†]. In addition, the outputs for each network were selected by grouping the variables based on the order they were stored. While this grouping may seem arbitrary, all simulation variables were stored next to variables with the same scientific units, as well as variables that represented very similar components within the simulation.

While we optimized the total number of FFNN models, we were not able to minimize the total number of linear models. The Lasso regression method used within our work is only able to approximate univariate response variables. This means we created a linear model per simulation output. This restriction results in using 80 linear functions to model the FG data set, and 90 linear functions to model the MO1 data set. While the overall model count is high, the overall training time scales very well using the ADMM optimization technique. In fact, the computation time scaled better than the FFNN models on average.

[†] The MO1 and MO2 data sets originally contained 96 outputs, but six output variables for all simulations never changed and were removed.

Using the previously mentioned data sets and our model, we conducted two experiments. The first experiment tests how accurately a model approximates E+ when only seven simulation variables are varied (FG data set). This allows us to estimate how sensitive the learned models are to fluctuations in the building parameter inputs by using a very densely sampled simulation input set. In this particular experiment, we selected the best FFNNs by testing models with 5, 10, and 15 hidden neurons, and we selected the best Lasso regression model by testing λ values between 0 and 1 using 0.15 increments. The training set contains 250 simulations and the testing set contains 750 simulations; we selected the models that performed best overall on the testing set.

The second experiment measures how well our models approximate E+ when presented with a very coarse sampling of the simulation input parameters. Using the MO1 data set, mentioned in Section 2.2.3, we train a FFNN and Lasso regression model. We tested the FFNN models with 5, 10, and 15 hidden nodes. For the Lasso regression models, we searched for the best λ value between 0 and 1 using 0.15 increments. We used 300 randomly sampled simulations from the slightly denser MO2 data set for testing and comparing both methods.

5.2.2 Performance Metrics

Within the building community, there are three accepted default metrics for comparing prediction accuracies – Coefficient of Variance (CV) (Kreider and Haberl, 1994), Mean Bias Error (MBE) (Kreider and Haberl, 1994), and Mean Absolute Percentage of Error (MAPE) (Edwards et al., 2012). This work uses the first metric, CV, to evaluate performance and does not utilize the other two common metrics. These metrics are not used because they are primarily used as tie breaking metrics within the building energy modeling community (Kreider and Haberl, 1994; Edwards

et al., 2012). The CV metric is defined as follows:

$$\text{CV} = \frac{\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}}{\bar{y}} \times 100 \tag{5.8}$$

where $\hat{y}_i$ is the predicted energy consumption, y_i is the actual energy consumption, and $\bar{y}$ is the average actual energy consumption. The CV value determines how much the overall prediction error varies with respect to the target's mean. A high CV score indicates that a model has a high error range.

In addition to the CV metric, we rely on the Root Mean Square Error metric (RMSE) to evaluate model prediction quality. The RMSE metric represents a model's average error rate per estimate. Additionally, the RMSE metric is a subcomponent used in computing the CV metric; it is the numerator in the CV metric equation (Eq 5.8). We use the RMSE error metric to facilitate broader comparisons, because the CV, MBE, and MAPE metrics are not scale invariant, which means these metrics can provide a false impression about model accuracy. For example, a target variable with a mean less than zero will produce very large metric values, even when the RMSE metric presents an acceptable error rate. The inverse problem applies to variables with large means as well; these metrics can produce low values even with a large error rate. However, the large mean scenario produces less overexaggerated metric measures, and is still reliable.

5.3 Results

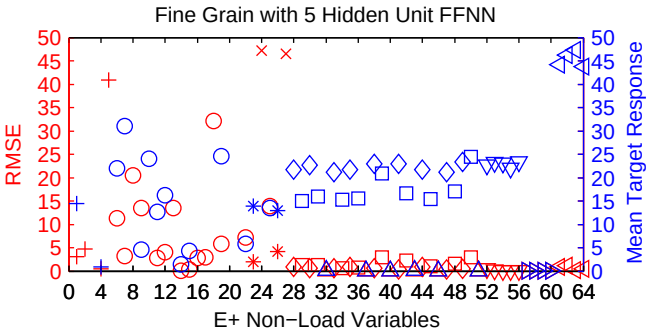
Both data sets contain a large number of output variables. This makes it difficult to present the variables in a table. Therefore, we have elected to use figures, rather than tables, to provide broader comparisons across the models. In addition, we have split the output variable figures into two groups. The first group depicts non-load variable results, while the second depicts load variables. All non-load variables are only referenced by a number rather than their actual name. However, we provide

explicit names for the load variables within this section to facilitate discussion in Section 5.4. In addition, we provide a detailed variable list in Appendix 6.10

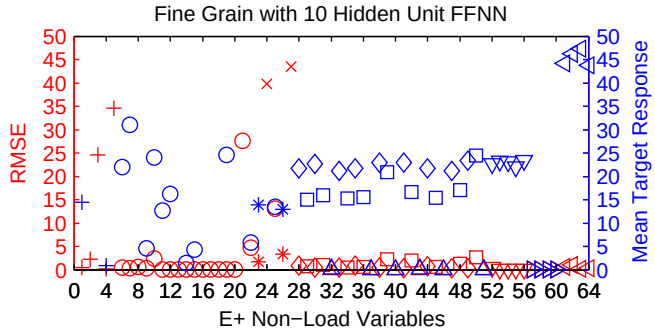
Our broad comparison figures do not directly present the CV metric. Rather, the figures present the two values used to compute the CV metric. The left y-axis represents the RMSE metric and right y-axis represents a response variables' mean (MTR). Normally, we would present the ratio between these two values, the CV metric, but several response variables' are small. This leads to misleading CV values that show poor relative performance whereas the absolute performance is good. When reading and interpreting these performance figures, the distance between the RMSE and MTR indicates overall performance. If the distance is large and the RMSE is below the MTR, then the prediction performance is excellent. However, if the RMSE is above the MTR, then prediction performance is poor. Finally, all non-load figures restrict the y-axis to $[0, 50]$ for RMSE and MTR, and the RMSE error values in all figures never exceed the range, but a few MTR values are either significantly beyond this range or hidden behind the figure's legend. This means if a corresponding MTR is not observed for an RMSE data point, then the MTR value's scale exceeds the current figure's range.

5.3.1 Fine Grain

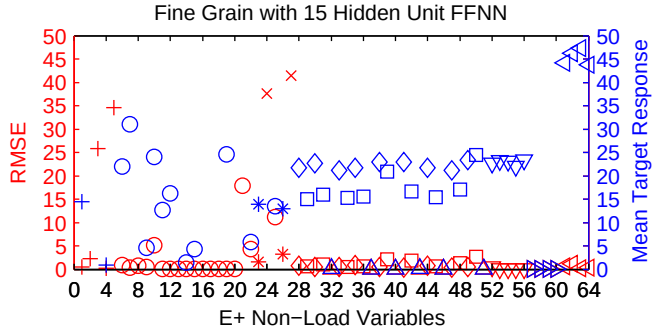
Figure 5.1 presents the FFNN non-load variable prediction results for our experiments on the FG data set. Most models perform the same on the response variables, but a few variables do present noticeable differences across the different models — in particular, the environmental and electrical variables between 1 and 16, as well as the envelope's heat gain and loss variables between 20 and 28. The variables between 1 and 16 present the best performance with 10 hidden units (Figure 5.1(b)). Analyzing the figure closely reveals that variable 10 has a much better error rate with 15 hidden units. Even though the performance for the other variables between 1 and 16 produce similar performance with 15 hidden units (Figure 5.1(c)), 10 hidden units is considered



(a) 5 Hidden Units



(b) 10 Hidden Units



(c) 15 Hidden Units

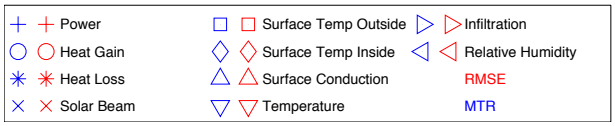


Figure 5.1: FFNN prediction results for the FG non-load variables with 5 (Figure 5.1(a)), 10 (Figure 5.1(b)), and 15 (Figure 5.1(c)) hidden units. Error bars are not presented to enhance figure readability, all standard deviations are less than 0.447.

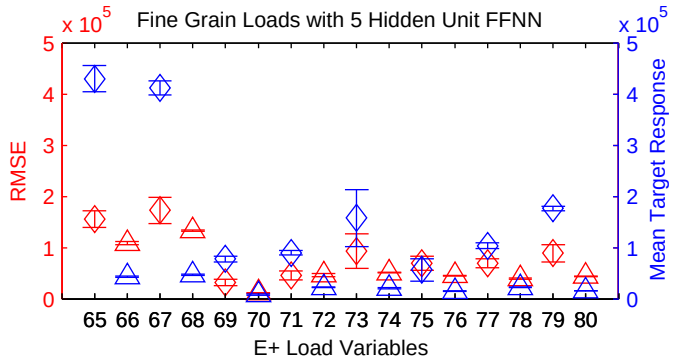
the better model since it takes less time to calculate and is more likely to generalize to new data.

The 15 hidden unit model presents the best performance on the variables between 21 and 28. These results suggest that the best approach for modeling the FG data set involves using the 10 hidden unit network to model all variables, except the ones between 21 and 28. Those variables should be modeled by the network with 15 hidden units. Our network design makes this simple, since there is a network per 10 output variables. This means either producing redundant estimates for variables 20 and 29 or allowing the 15 hidden unit network to estimate these variables as well. The secondary option is sufficient considering the 10 and 15 hidden unit networks' performance on 20 and 29 is similar.

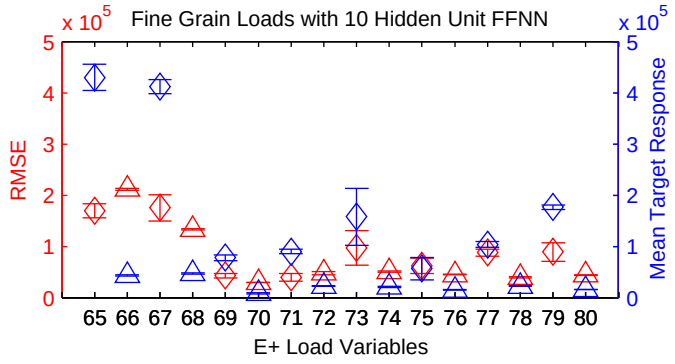
While we consider the non-load prediction performance excellent, the FFNN load prediction performance shows room for improvement. The load variables in Figure 5.2 represent the Sensible Heat, Latent Heat, Sensible Cooling, and Latent Cooling for four different zones in the following order: living room (LR – variables 65-68), master bedroom (MB – 69-72), basement (BM – 73-76), and second floor (SF – 77-80).

Analyzing Figures 5.2(a) and 5.2(c) show that the 5 and 15 hidden unit models present the best prediction results overall. However, the 5 hidden unit model has the least variance within its RMSE error, which means this model is more stable than the 15 hidden unit model. In addition, the 5 hidden unit model has less model complexity. Therefore, the 5 hidden unit FFNN is considered best for predicting the FG load variables.

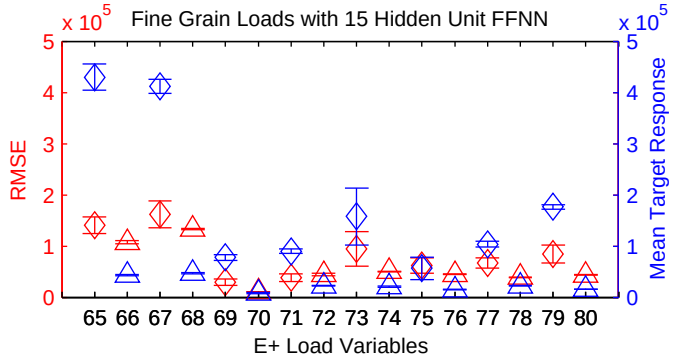
Figure 5.3 presents the results from testing the linear model on the FG data set. The model performs well on the non-load variables and is fairly competitive with the previous FFNN models. However, some error rates have much higher variance than the best FFNN model (Figure 5.1(c)) for variables 7 and 20-28. In addition, all error rates for these variables are statistically worse than the FFNN error rates. This means these variables have a nonlinear relationship with their inputs, and the Lasso regression is not as capable a methodology for capturing these patterns.



(a) 5 Hidden Units



(b) 10 Hidden Units



(c) 15 Hidden Units

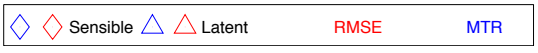


Figure 5.2: FFNN prediction results for the FG load variables with 5 (Figure 5.2(a)), 10 (Figure 5.2(b)), and 15 (Figure 5.2(c)) hidden units.

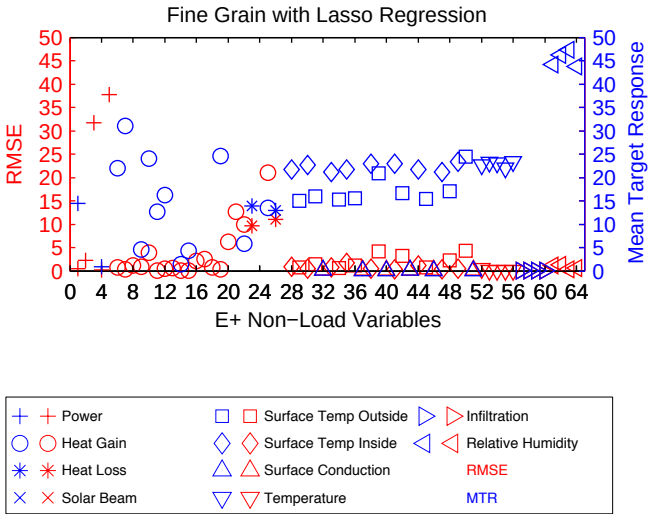


Figure 5.3: Lasso regression model's performance on the FG data set's non load variables. Error bars are not presented to enhance figure readability, most standard deviations are less than 1. However, variables 5, 22, 23, 25, and 26 have standard deviations between 2 and 6. Variable 24's RMSE is 84.45±22.4921 and its MTR is 122.56±0.00.

While the other non-load variables are all statistically worse than the FFNN models, the Lasso method only uses an input subset[‡] to make all the predictions, which means the linear model is using less information than the FFNN. In addition, most variables are only marginally worse, e.g., 0.2 - 5.0 difference in RMSE. However, a few variables differ in RMSE by 20 or more. This implies that the Lasso regression method is fitting simpler models by reducing the number of input variables used within the model, which results in a simpler model, but not always a better performing model. However, simplistic models are more likely to generalize to unseen events. In addition, the Lasso models learn much faster than the FFNNs, as previously discussed in Section 5.1.4.

The Lasso regression load prediction results (Figure 5.4) are very similar to the FFNN results (Figure 5.2). While the Lasso regression results are worse, the model

‡ The subset refers to the inputs that have a non-zero weight in the model.

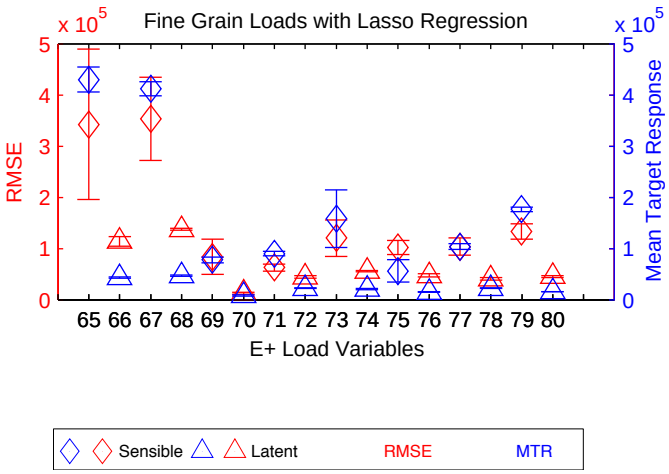


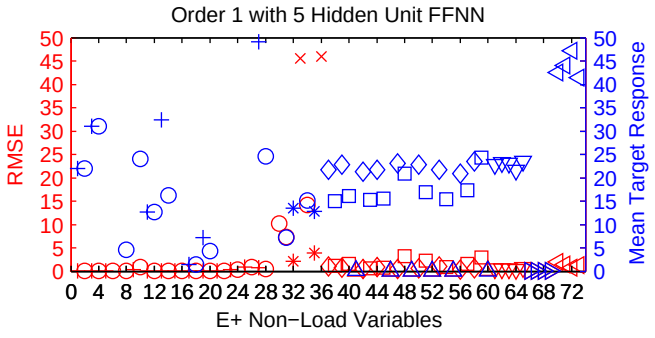
Figure 5.4: Lasso regression model's performance on the FG data set's load variables.

performed best on the same variables that the FFNN models were able to predict – variables 65, 67, 71, 73, 77, and 79. However, the other load variables were not fit well by either method, which implies that there is not sufficient information within the raw data set to predict the other load variables.

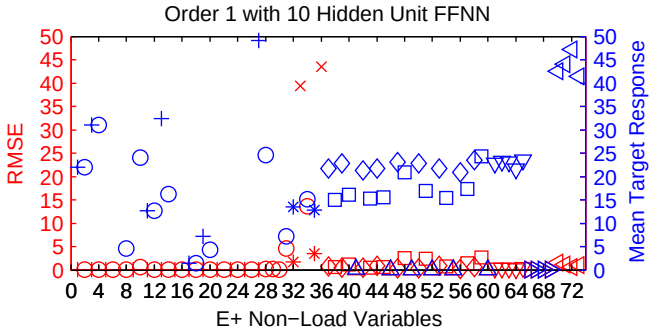
5.3.2 Markov Order 1 & 2

Our experiments with the MO1 and MO2 data sets are more challenging than the experiments with the FG data set, because we are testing how well our models generalize when trained with a limited representation of the input space, i.e., trained using the entire sparse MO1 data set. The extra difficulty is introduced by testing with sampled MO2 simulations, which contain input combinations that will never appear within the training set and may lead to very inaccurate predictions.

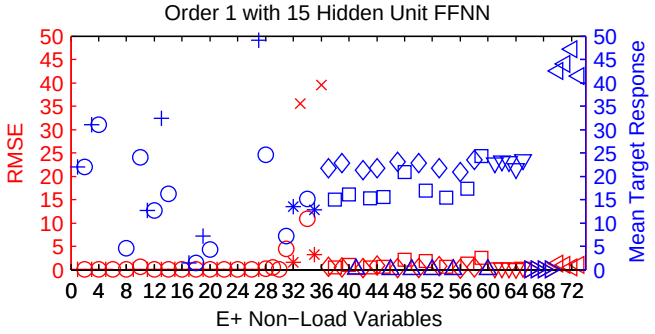
Figure 5.5 presents the MO1 experimental results for non-load variables and Figure 5.6 presents the MO1 experimental results for the load variables. Prediction for non-load variables is slightly better than the results presented on the FG data set, but this is primarily attributed to adding additional equipment related output variables,



(a) 5 Hidden Unit



(b) 10 Hidden Unit



(c) 15 Hidden Unit

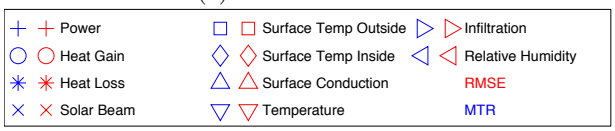


Figure 5.5: FFNN prediction results for MO2 non-load variables with 5 (Figure 5.5(a)), 10 (Figure 5.5(b)), and 15 (Figure 5.5(c)) hidden units. Models were trained using all MO1 data. Error bars are not presented to enhance figure readability, variable standard deviations are less than or equal to 1.21. Only variable 10 has a larger standard deviation, 6.86.

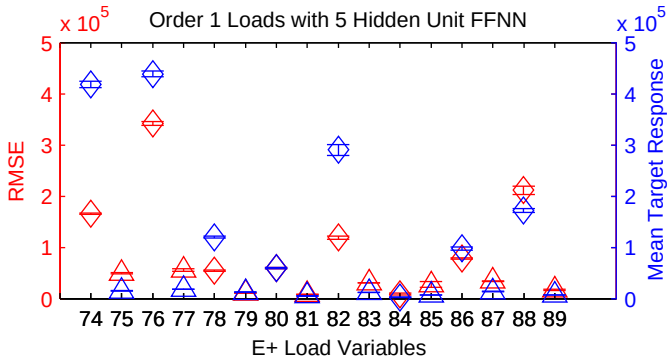
which are modeled efficiently by the operation schedule. All models produce about the same performance results on the non-load variables. However, the model with 15 hidden units performs stastistically better than the other models on variables 32 through 36. This means the 15 hidden unit FFNN best models the non-load variables.

Similar to the FG data set, load variables remain difficult to predict and require further analysis to improve their prediction accuracy. Additional analysis and improvement directions are discussed in Section 5.4. However, the FFNN models were able to predict variables 74, 76, 78, and 82 (Figure 5.6). However, we observed that the 10 hidden unit FFNN produced the best performance on the MO2 data set. We based this conclusion on variables 78, 82, and 86's different performance between the two models.

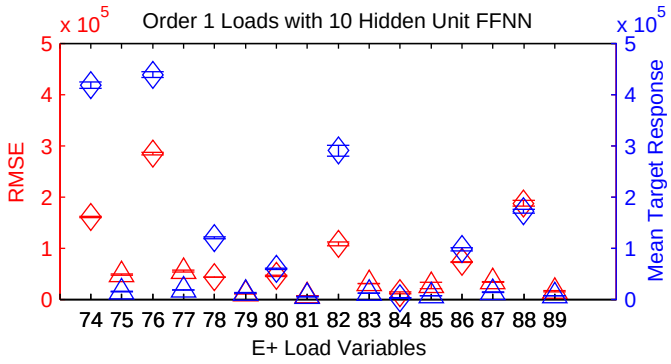
The Lasso regression results for the MO2 non-load variables are shown in Figure 5.7. These results illustrate that many variables within the MO1 and MO2 data set can be modeled using linear models. In addition, it highlights the variables that require a nonlinear model as can be seen in Figure 5.5(c). The variables between 6 and 12 as well as 28 and 36 are fit better by the FFNN model.

However, both models present a high variance on variable 10. The high variance is directly associated with the sparse parameter sampling found in the MO1 data set. This particular variable has instances where it produces different response behavior, as seen in Figure 5.8. The response behavior completely differs in scale for a few simulations used within the test data set. This means that coarse parameter sampling used to generate MO1 may have limited abilities to produce meaningful general purpose models. However, it also implies that models created from the MO2 data set will have similar deficiencies, because a limited sampling can only represent a fraction of the entire domain's behavior. Ultimately, we would like to explore simulation parameter sampling strategies that consider the learner's fitting capabilities as well as implement methods that can estimate the variance associated with each prediction.

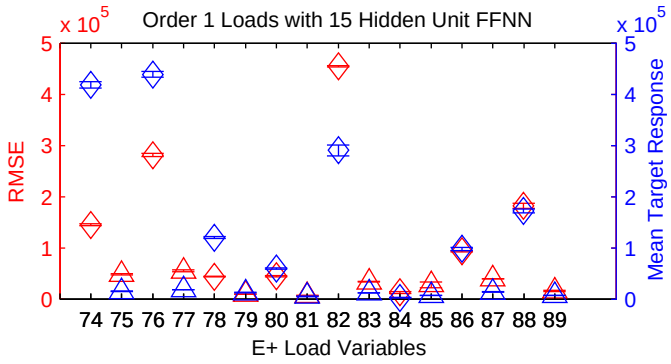
Similar to the FG data set results (Figure 5.4), the load predictions with Lasso regression (Figure 5.9) are all worse than all the best FFNN results (Figure 6.7(b)).



(a) 5 Hidden Unit



(b) 10 Hidden Unit



(c) 15 Hidden Unit

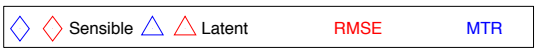


Figure 5.6: FFNN prediction results for the MO2 load variables with 5 (Figure 6.7(a)), 10 (Figure 6.7(b)), and 15 (Figure 6.7(f)) hidden units. Models were trained using all MO1 data.

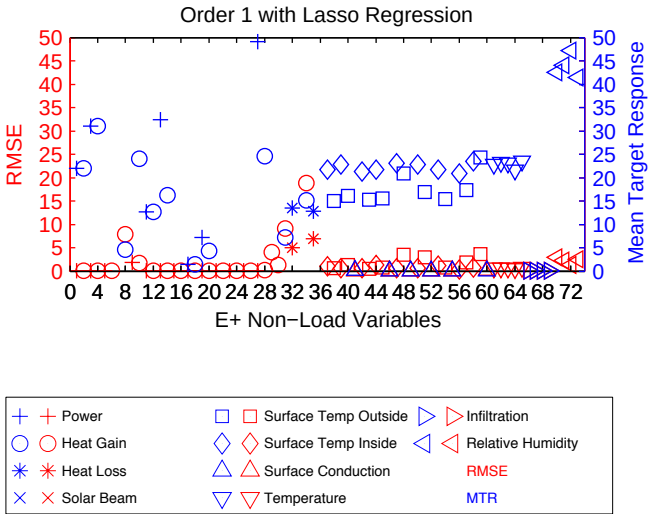


Figure 5.7: Lasso regression model's performance on the MO2 data set's non-load variables. The model was trained using all MO1 data. Error bars are not presented to enhance figure readability, variable standard deviations are less than or equal to 1.20. Only variable 10 has a larger standard deviation, 6.47.

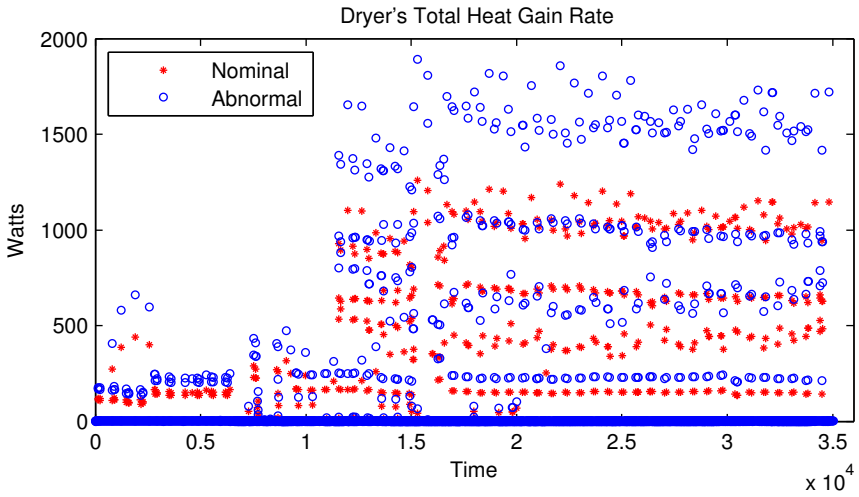


Figure 5.8: Compares the average MO2 dryer heat gain response against an observed scale shifted response. Two other MO2 test simulations present the same scale shift response behavior.

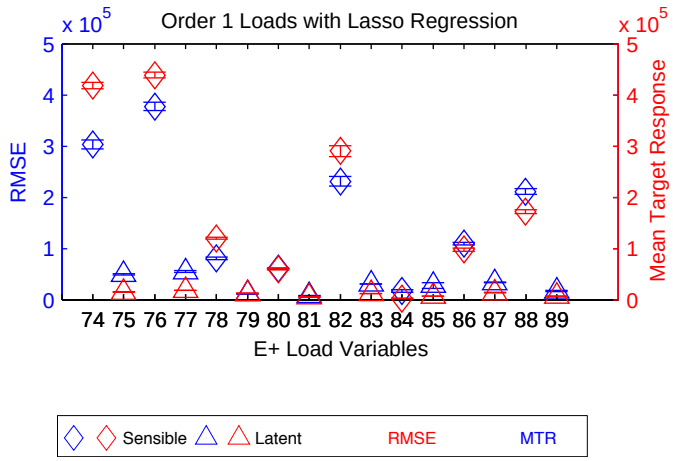


Figure 5.9: Lasso regression model's performance on the MO2 data set's load variables. The model was trained using all MO1 data.

Table 5.1: Illustrates all model's Whole Building Energy consumption (MO2 Variable 90) prediction accuracy. Several standard deviations were very small, making them essentially zero.

Model	RMSE (W)	MEAN (W)	CV (%)
FFNN 5	9.125±0.00	1756.8±0.00	0.519±0.00
FFNN 10	1.164±0.00	1756.8±0.00	0.066±0.00
FFNN 15	1.061±0.00	1756.8±0.00	0.060±0.00
Lasso	4.797±1.61	1756.8±0.00	0.273±0.09

However, the Lasso regression method was able to fit the same load variables as the FFNN model – variables 74, 76, 78, and 82. This result continues to imply that there is not enough information to model the other load variables using the raw data set.

The MO2 data set's 90th simulation output variable represents Whole Building Energy (WBE) consumption, which is not present in the FG data set. Table 5.1 presents the WBE prediction results for all models. These results illustrate that the Lasso model provides a better fit than the 5 hidden node FFNN model, but provides a worse fit than the 10 and 15 hidden node models. While the Lasso model does not perform as well, its overall training time is substantially better than the

Table 5.2: The first column represents the single MO1 model training time, and the second column is the necessary time to train all MO1 models in serial. The prediction time represents single model execution time.

Model	Training Time (Hr)	Total Training Time (Hr)	Prediction Time (sec)
FFNN 5	∼2	∼18	∼2.70
FFNN 10	∼8	∼72	∼2.85
FFNN 15	∼24	∼216	∼2.93
Lasso	∼0.2833	∼25.50	∼2.90

FFNN 10 and 15 hidden node models (Table 5.2). These performance characteristics indicate that it is best to use the Lasso regression model to predict all variables that can be sufficiently represented using a linear model. This is especially true when one has sufficient computational resources to run the learning algorithms in parallel. The Total Training Time represents the execution time associated with training each individual model in serial. A more parallel approach will converge to the single model training time. Lastly, the overall prediction time represents the parallel execution speed for running the nine MO1 FFNN models in parallel and running the entire Lasso regression model as a single matrix multiplication. This indicates that the Lasso regression method's testing speed scales better than the FFNN when parallel execution is not possible.

5.4 Discussion

Several interesting findings have been observed in regards to both data sets and prediction accuracy. First, we have observed a simulation clustering phenomenon, which may provide insight into E+ approximation efforts, and illustrates how prediction accuracy varies as a function of simulation data. Second, we discuss HVAC schedule features for improving overall heating and cooling load predictions.

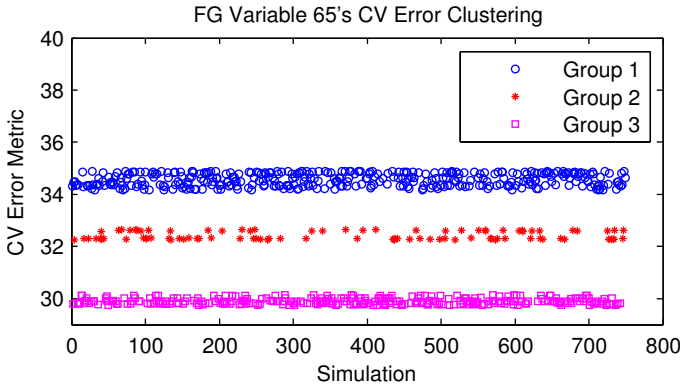
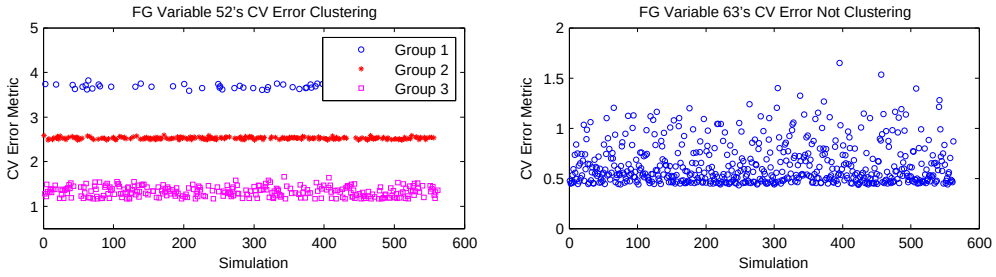


Figure 5.10: Illustrates the FFNN model's CV error clustering into distinct clusters on the Fine Grain dataset.



(a) Lasso Cluster

(b) Lasso No Cluster

Figure 5.11: Illustrates that the Lasso regression models can produce distinct clusters when a linear model captures the full relationship between inputs and outputs (Figure 5.11(a)). When a nonlinear model is required, Lasso regression fails to produce the distinct clustering (Figure 5.11(b)).

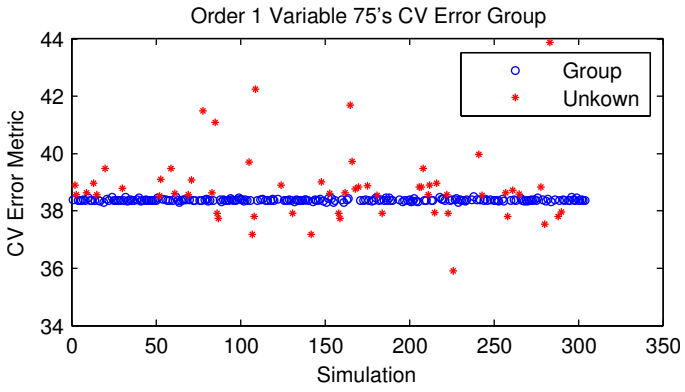


Figure 5.12: Illustrates the FFNN model's CV error clustering into a distinct cluster on the Markov Order 2 dataset.

Through this work we have noticed several interesting properties exhibited by the predictive models. For example, the CV error metric measured for FFNN prediction on FG data set's variable 65 constructs well defined clusters (Figure 5.10). These well defined clusters are created by the FFNN and Lasso regression models for multiple variables. However, the Lasso regression model does not always exhibit the clustering behavior; it only occurs if the variable is sufficiently well predicted by a linear model (Figure 5.11(a) and 5.11(b)). Neither model exhibits the same clustering behavior in the MO1 experiments. The MO1 experiments show a single group (Figure 5.12). This clustering property suggests that, as we increase the E+ parameter sampling density (i.e., sample the data more finely), we will most likely see the number of clusters increase. This is even more likely if we expand on the total number of parameters that are sampled. In such a case, ensemble learning for specialization in predicting each cluster may be fortuitous.

Another important observation is the fact that each cluster, representing simulations that are equally misinterpreted by the predictive model, means the best E+ approximation model should be built for each cluster. For example, in Sections 3.7 and 3.8, we discuss two methods for predicting future hourly residential electrical consumption via clusters determined by C-means and Hierarchical Mixture of Experts. While our results here imply these model types will produce better

performance on densely sampled simulations, this performance increase has not yet been quantified through experimentation, due to the scalability improvements necessary for handling such large data.

In particular, the clustering process needs to group the E+ data by simulation. However, this is a difficult problem due to the curse of dimensionality Hinneburg et al. (1999), which in this instance means that producing accurate clustering becomes increasingly difficult as dimensionality increases. Each simulation contains 35,040 simulation output vectors. The FG data set contains 80 outputs while the MO1 data set contains 90 outputs. This makes it possible for typical clustering metrics (e.g., Euclidian distance) to produce meaningless similarity measures. The more appropriate approach is clustering the individual output vectors, establishing individual cluster membership probabilities based on each simulation's distribution across the clusters, and constructing the final model cluster centroids using the membership probabilities and the input vectors[§]. While these various challenges illustrate that approximating E+ or large data is difficult, it establishes many directions for future research within the machine learning field and building spaces community. We discuss these directions in more detail in the future work section.

In an effort to improve overall heating and cooling load predictions, we added features related to HVAC operation schedule and temperature gradients to the input set. The temperature gradient features include the inside and outside temperature gradient. The inside temperature gradient represents the average temperature change across the building zones. In our experiments the building zones correspond to the living room, master bedroom, basement, and second floor. The outside temperature gradient represents the change in dry bulb temperature. Using these temperature gradient features, we manually constructed a heuristic indicator function that estimates whether the HVAC is currently heating, cooling, or neither. The indicator function first uses the time and region information to determine whether

§ All clusters must ultimately relate to the input vectors, because testing a simulation parameter setting does not have corresponding output.

heating or cooling loads are realistic[¶]. The function allows heating to be active only during October through March and allows cooling to be active during the remaining months. Finally, we use the gradient direction to estimate the on or off state. If the inside gradient is increasing and the outside gradient is decreasing, then the heat is activated, provided the time corresponds with a heating month. The inverse is used to establish when the cooling is active.

Using these features and the FFNN model, we repeated the FG and MO1 experiments on the heating and cooling load variables. The FG load results are shown in Table 5.4. Analyzing the FG table illustrates that the HVAC operation features and temperature gradient features produce statistically better prediction results with 95% confidence on the LR, MB, and BM sensible heating loads (variables 65,69, and 73) as well as LR's latent heating and cooling loads, and MB's latent cooling load (variables 66, 68, and 72). Lastly, the LR's sensible cooling load and MB's latent heating load were unchanged (variables 67 and 70). All other FG variable predictions are worse.

The MO1 experiments (shown in Table 5.4) with the HVAC operation and temperature gradient features produce statistically better LR, BM, and Second Floor (SF) sensible heating load predictions (variables 74, 82, and 86). In addition, the features produce statistically better MB latent heating load predictions (variable 79). The load predictions for variables 75, 77, 81, 83, 85 and 89 were not statistically different. All other load predictions were statistically worse (variables 76, 78, 80, 84, 86, 87, 88).

Incorporating these additional features into the learning systems clearly provides mixed results since not all load predictions improved. In addition, the improved variables did not reach prediction rates similar to the better sensible load predictions (e.g., LR's sensible heating and cooling). Incorporating these findings with the Lasso regression results from Section 5.3, which suggest that necessary information for

[¶] The simulations in all experiments use a constant set point for the entire year.

Table 5.3: Compares the best FG FFNN model results, without HVAC features, against the best FG FFNN model with HVAC operation features. Both models use 15 hidden units. Variables that show improvement are marked in blue and those show degradation are marked in red.

Variable	RMSE (#×10^5 J)	MEAN (#×10^5 J)	CV (%)
65-Old	1.4022±0.1624	4.2998±0.2505	32.522±2.175
65-New	1.3596±0.1721	-	31.516±2.420
66-Old	1.0741±0.0334	0.4286±0.0134	250.657±4.533
66-New	1.0618±0.0333	-	247.796±4.536
67-Old	1.6204±0.2621	4.1195±0.1343	39.170±5.088
67-New	1.6216±0.2683	-	39.195±5.238
68-Old	1.3304±0.0158	0.4729±0.0111	281.406±3.914
68-New	1.3182±0.0148	-	278.826±4.124
69-Old	0.2922±0.0624	0.7780±0.0539	37.366±6.505
69-New	0.2764±0.0652	-	35.308±6.922
70-Old	0.1081±0.0043	0.0816±0.0076	133.239±9.119
70-New	0.1075±0.0042	-	132.564±9.041
71-Old	0.3848±0.0742	0.8996±0.0404	42.490±6.344
71-New	0.3913±0.0789	-	43.190±6.835
72-Old	0.4452±0.0295	0.2237±0.0050	198.806±8.934
72-New	0.4374±0.0309	-	195.306±9.626
73-Old	0.9445±0.3328	1.5752±0.5593	63.423±21.675
73-New	0.9123±0.3572	-	60.302±19.348
74-Old	0.5032±0.0064	0.2076±0.0082	242.661±8.171
74-New	0.5129±0.0064	-	247.333±8.410
75-Old	0.6210±0.1473	0.5637±0.2172	131.310±87.927
75-New	0.6621±0.1437	-	137.680±81.744
76-Old	0.4475±0.0061	0.1470±0.0032	304.523±4.892
76-New	0.4495±0.0060	-	305.913±4.947
77-Old	0.6698±0.0972	1.0371±0.0537	64.311±6.454
77-New	0.6681±0.1022	-	64.131±6.947
78-Old	0.3876±0.0105	0.2231±0.0092	173.867±5.115
78-New	0.3925±0.0105	-	176.058±5.192
79-Old	0.8449±0.1765	1.7647±0.0457	47.654±8.738
79-New	0.9044±0.1788	-	51.024±8.785
80-Old	0.4372±0.0070	0.1573±0.0037	277.937±2.613
80-New	0.4366±0.0068	-	277.518±2.708

Table 5.4: Compares the best MO2 FFNN model results, without HVAC features, against the best MO2 FFNN model with HVAC operation features. Both models use 10 hidden units. Variables that show improvement are marked in blue and those that show degradation are marked in red.

Variable	RMSE (#×10^5 J)	MEAN (#×10^5 J)	CV (%)
74-Old	1.6084±0.0165	4.1829±0.0670	38.461±0.654
74-New	1.5322±0.0173	-	36.641±0.723
75-Old	0.4831±0.0132	0.1470±0.0048	328.746±9.685
75-New	0.4815±0.0130	-	327.704±9.755
76-Old	2.8421±0.0290	4.3868±0.0512	64.792±0.6371
76-New	2.9108±0.0349	-	66.356±0.605
77-Old	0.5497±0.0229	0.1812±0.0049	303.245±8.912
77-New	0.5519±0.0226	-	304.489±8.8140
78-Old	0.4313±0.0056	1.1951±0.0175	36.092±0.597
78-New	0.4653±0.0068	-	38.934±0.557
79-Old	0.1044±0.0026	0.1238±0.0043	84.380±1.939
79-New	0.1032±0.0025	-	83.396±2.042
80-Old	0.4558±0.0187	0.5947±0.0091	76.640±1.551
80-New	0.4713±0.0129	-	79.241±1.679
81-Old	0.0665±0.0021	0.0512±0.0017	130.159±5.149
81-New	0.0664±0.0021	-	129.885±5.121
82-Old	1.0850±0.0371	2.902±0.1096	37.476±2.840
82-New	1.0372±0.0371	-	35.820±2.678
83-Old	0.3082±0.0025	0.1178±0.0021	261.775±8.171
83-New	0.3080±0.0026	-	261.556±2.642
84-Old	0.1228±0.0155	0.0225±0.0071	564.939±111.943
84-New	0.0798±0.0175	-	364.840±81.744
85-Old	0.2670±0.0590	0.0693±0.0038	383.244±47.206
85-New	0.2669±0.0589	-	383.040±47.123
86-Old	0.7264±0.0101	0.9751±0.0256	74.531±1.813
86-New	0.6800±0.0114	-	69.761±1.483
87-Old	0.3379±0.0038	0.1372±0.0023	246.346±3.786
87-New	0.3405±0.0037	-	248.260±3.814
88-Old	1.8742±0.0609	1.7166±0.0337	109.169±2.408
88-New	1.9058±0.0666	-	111.002±2.753
89-Old	0.1583±0.0111	0.0728±0.0013	217.361±14.703
89-New	0.1600±0.0111	-	219.708±14.657

predicting latent loads is missing, shows that improving overall load predictions is a challenging problem when relying on only the building envelope parameters, operation schedule, and weather information to make predictions. This leaves two directions – 1) Bottom-up feature synthesis from existing data; or 2) Top-down analysis, through continued interaction with domain experts, to determine additional E+ information that could improve approximations.

Figures 5.13(a) and 5.13(b) show our HVAC heating and cooling (on/off) features and the MO1 latent cooling and heating loads for the LR zone. Figure 5.13(a) shows that the HVAC heating is on mostly when the latent loads are non-zero and similarly for latent cooling loads in Figure 5.13(b). Our current HVAC features correlate well with the MO1 sensible and latent loads, so improvement in a model's prediction accuracy could only be achieved through additional information.

However, Figures 5.14(a) and 5.14(b) illustrate that the FG LR latent loads are uniformly distributed throughout the year. These latent loads are not indicative of the HVAC's operation. Identifying and recording the necessary information will require executing additional simulations. In addition, we need to ensure that all information gathered from the E+ internals is used in a repeatable manner, i.e., the information is not used in such away that we can not use the models to provide E+ approximations for other buildings. This issue is discussed further in future work.

5.5 Results Summary

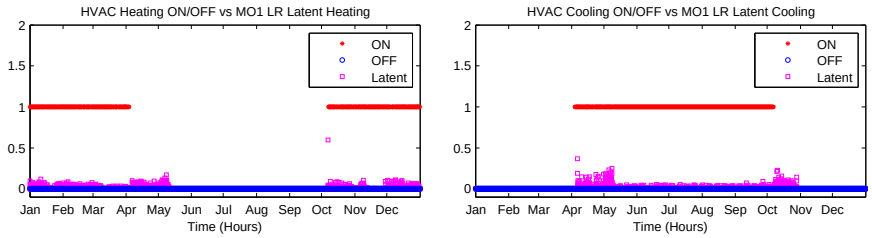
Using FFNN and Lasso regression, we demonstrated the ability to produce E+ approximation models for a residential building. Our models use building envelope parameters selected by domain experts, an operation schedule, and weather data. These models are able to successfully predict a majority of the domain expert selected output variables. We are able to identify which output variables require a nonlinear model, based on comparing the FFNN and Lasso models directly. However, these

models only have moderate success at predicting sensible heating and cooling loads, and are unsuccessful at predicting the latent cooling and heating loads.

In an effort to improve the load predictions, we incorporated HVAC operating heating and cooling features, which indicates the on and off states for these respective operating conditions. These new features present mixed results. Some load predictions show improvement, while others remain the same or diminish. Based on these results and Lasso regression's ability to automatically select relevant inputs, we concluded that either better use of existing information or additional information is necessary to better predict the latent load variables. We continue to analyze additional features and internal E+ variable information that can be incorporated into our prediction process without diminishing the E+ approximation's generality.

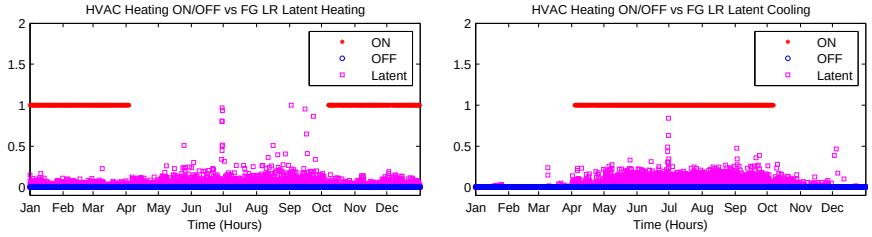
The Lasso model is able to predict an entire yearly simulation in ∼3 seconds, and the FFNN models can achieve the same execution time when run in parallel. These runtimes are considerably faster than the average E+ runtime (∼2-3 minutes). This performance increase will provide improvement to the overall building calibration process, when using the well predicted variables in the tuning objective.

Lastly, the three data sets (FG, MO1 and MO2) allowed us to determine that the best E+ approximation model requires multiple models, as discussed in Section 5.4, which can tailor the learning to individual cluster attributes.



(a) MO1 LR Latent Heating (b) MO1 LR Latent Cooling

Figure 5.13: The HVAC on and off operating feature overlayed onto a sample MO1 LR latent heating (Figure 5.13(a)) and sample MO1 LR latent cooling (Figure 5.13(b)).



(a) FG LR Latent Heating (b) FG LR Latent Heating

Figure 5.14: The HVAC on and off operating feature overlayed onto a sample FG LR latent heating (Figure 5.14(a)) and sample FG LR latent cooling (Figure 5.14(b))

Chapter 6

Learning Simulation Variable Relationships

Tuning the EnergyPlus (E+) simulation model requires that one understands how all the input variables interact and the effects these variables have on the simulation output. This knowledge is clearly required for tuning the E+ input parameters, because improving the simulation quality requires understanding how to adjust input parameters to improve the output. In addition, approximating the E+ simulation clearly requires understanding the relationships between the input and output variables, because a model that approximates E+ must be able to map a provided set of inputs to the correct set of output variables. Therefore, fundamentally tuning and approximating E+ both require solving the same core problem: "Determine a model that describes the relationships between the E+ variables".

If one views the E+ input and output variables as a set of random variables, e.g., $\{X_1, X_2, ..., X_N\}$, then learning the relationship between the variables can be formulated as learning the joint probability distribution, $P(X_1, X_2, ..., X_N)$, over these variables. This means that the true joint distribution defines a probabilistic surface over the random variables, which implies that the joint distribution represents all the complex relationships between the E+ variables. The joint distribution can be

used to tune E+ models, and likewise it has the ability to approximate the E+ simulation. The tuning problem can be solved by using reference output data or real world sensor data to find the input variables that maximize the posterior probability. In mathematical notation, one wants to maximize the following probability:

$$\Pi_{i=1}^{N} P(X|Y_i)P(X) \tag{6.1}$$

where X denotes the E+ building specification variables and Y denotes the observed output data or real world sensor data, weather data, and operation schedule. This equation assumes that all observations are independent and identically distributed. Applying Bayes' Theorem allows 6.1 to be written as:

$$\Pi_{i=1}^{N} P(X, Y_i) \tag{6.2}$$

which is the joint probability distribution. Conversely, the approximation process requires maximizing the following posterior probability:

$$\Pi_{i=1}^{N} P(Y_i|X)P(Y_i) \tag{6.3}$$

where X represents the inputs and observed operation schedule, and Y_i represents a single output sample. This approximation method requires finding multiple samples that maximize the posterior probability, rather than a single assignment. However, forward approximating E+ using this method is computationally slower than using our surrogate models. Therefore, we are only focusing on estimating building parameters and only present the other capability for completeness.

The remainder of this chapter is organized as follows: Section 6.1 provides background information on Probabilistic Graphical Models (PGMs); Sections 6.2, 6.3, and 6.4 provide a comprehensive review on structure learning methods; Section 6.5 presents our approach; Section 6.6 presents our experimental setup; Sections 6.7.1,

6.7.2, and 6.7.3 presents our experimental results; Section 6.8 provides discussion; and Section 6.9 summarizes all findings within this chapter.

6.1 Probabilistic Graphical Models

Learning the true joint probability distribution directly in the general case is computationally intractable, especially when dealing with a large number of continuous random variables. In general, it is assumed the joint probability factorizes into several smaller more manageable computational problems. The factorization is based purely on conditional independence and is represented using a graphical model G, where G is a graph with V nodes representing random variables and E edges. The edges in E represent conditional dependencies between the random variables in the graph G. The graph structure is intended to represent the true factorization for the joint probability distribution into simpler components. These types of models have been applied to many fields and produced great results, such as topic classification Blei et al. (2003), document classification Bernardo et al. (2003), activity recognition Zhang and Parker (2011), disease diagnosis Jordan et al. (1999), and many more.

The graphs used to represent factorized joint probability distributions are either direct acyclic graphs (DAG), or they have bidirectional edges. The first form assumes that the graph represents a Bayesian Network and the latter form assumes that the graph represents a Markov Network. These graph types both assume that a joint probability distribution factorizes according to their structure. However, a Bayesian Network assumes a much simpler probabilistic dependency between the variables, in which a variable is conditionally independent of all other variables given its parents. On the other hand, a Markov network assumes variables X are conditionally independent from variables Y provided that they are d-separated by variables Z. X and Y are said to be d-separated by Z if and only if there does not exist a path from the variables in X to the variables Y that does not pass through the variables Z Koller and Friedman (2009). Clearly, the Bayesian Network makes stronger assumptions about

independence, but these assumptions make it easier to perform exact interference. In contrast, Markov Networks use approximate inference methods, such as Loopy Belief Propagation. Even though a Markov Network is more computationally expensive than a Bayesian Network, it is a more preferred graph structure because it has much more representational power within the model. Therefore, a Markov Network should be preferred over a Bayesian Network, if it is computationally feasible to use one to approximate the joint distribution for the E+ variables.

While these graphical models are able to adequately represent joint distributions, the graph structures are generally predetermined. Given the total number of random variables within an E+ ($\sim$300 for our experiments) simulation, it is not feasible to specify the single best graph structure in advance. Therefore, algorithmic techniques must be used to find the best graph structure that matches the true joint probability distribution without overfitting the observed training samples. There are three categories of algorithms that aim to solve this problem, two of which are conventional methods. The first conventional method is Score and Search (Section 6.2), and is generally used to learn Bayesian Network structures. The second conventional approach is Constraint Based (Section 6.3) methods, which have been used to learn Bayesian and Markov Networks. The unconventional method uses strong probability assumptions combined with regression based feature selection methods to determine dependencies among the random variables (Section 6.4).

6.2 Score and Search

Score and Search algorithms try to search through the space of all possible models and select the best seen model. The best model is selected by using a global criteria function, such as the likelihood function. However, the most common criteria function is the Bayesian Information Criteria (BIC) Schwarz (1978):

$$\text{BIC(Data; G)} = \mathcal{L}(Data; G, \theta) + \frac{\log M}{2} * Dim[G] \tag{6.4}$$

where $Dim[G]$ represents the number of parameters estimated within the graph, M is the total number of samples, and $\mathcal{L}$ is the the log-likelihood function.

These Score and Search methods are generally used to try and find the best structure for Bayesian Networks, because the log-likelihood function factorizes into a series of summations. This factorization makes it very easy to modify an existing Bayesian Network and compute the modified score without recomputing the entire score. For example, adding an edge to an existing Bayesian network requires computing the new and previous conditional probability involving the child of the new edge, and adding the difference to the previous BIC score. Updating the penalty term is achieved by simply adding $N\frac{\log M}{2}$ to the updated BIC score, where N is the number of newly estimated parameters.

The most common method for performing Score and Search within the literature is Greedy Hill Climbing Russell and Norvig (2010). The algorithm starts with a candidate graph and explores valid augmentations to the graph. These augmentations include deleting edges, adding edges, and changing the direction for edges. The best valid augmentation is selected according to the defined criteria function, generally BIC, and a new candidate graph is generated. Note that an augmentation is only valid if the resulting candidate graph is a valid Bayesian Network. This greedy search approach is able to guarantee a locally optimal solution that will maximize the criteria function, but could be far away from the true model.

There are two algorithms that extend the basic greedy hill climbing algorithm by constraining the network search space, which allows for better solutions. The first algorithm is the Sparse Candidate method Friedman (1999). This method assumes that random variables that have a high measure of mutual information should be located closer to each other in the final network than variables with low mutual information. The mutual information for two discrete random variables, X and Y, is defined as follows:

$$\mathcal{I}(X,Y) = \sum_{x,y} P(x,y) log\left(\frac{P(x,y)}{P(x)P(y)}\right) \tag{6.5}$$

In the case of continuous random variables, the summation is replaced by integration.

In addition to using the mutual information within the data to restrict the search, the method restricts the total possible number of parents to a user specified value k. Combining the restricted number of parents with the mutual information criteria, the greedy algorithm selects the best candidate parent set for each random variable. Using the candidate parent set, an approximate Bayesian network is constructed. The network is approximate, because it may not actually be a valid Bayesian network. Standard greedy hill climbing is then applied to the approximate Bayesian network, but the valid augmentations are now restricted according to the best candidate parent set.

The Sparse Candidate algorithm is able to scale to large Bayesian Networks with hundreds of random variables, but selecting the correct value for k has a large impact on the approach and can greatly reduce solution quality if set incorrectly Tsamardinos et al. (2006). In addition, the method assumes that computing $\mathcal{I}(X, Y)$ is feasible through sampling in the discrete case with roughly 1000 samples Friedman (1999). However, approximating the $\mathcal{I}(X, Y)$ in the continuous case may not be feasible with so few samples, may be too computationally expensive, or produce poor approximations. In the latter case, greedy hill climbing may search a model space that does not contain the true candidate model resulting in a misspecified model.

The second algorithm, Max-Min Hill-Climb (MMHC Tsamardinos et al. (2006)), extends the basic greedy hill climbing algorithm by using a Constraint Based algorithm (Section 6.3) called Max-Min Parents and Children (MMPC Tsamardinos et al. (2003)) to determine the underlying undirected structure for each random variable. Given the approximate optimal substructure per variable, the algorithm proceeds to apply greedy hill climbing to find the DAG that maximizes the criteria function. However, edges can only be added to the graph if they follow the constraints specified by the undirected model. The advantage that this algorithm provides over the Sparse Candidate algorithm is a tighter upper bound on the run time; this algorithm also removes the parent restriction. However, the algorithm replaces the

parent restriction with a restriction on the size of the subsets that will be used for the MMPC's conditional independence testing. The algorithm's run time is $\mathcal{O}(|V|^2|PC|^{l+1})$ where $|V|$ represents the total number of random variables, $|PC|$ represents the largest set of parents and children, and l represents the maximum subset size allowed for conditional independence testing.

There are many other search methods that have been applied to try and find the best structure, such as genetic algorithms Larrañaga et al. (1996), best first search Russell and Norvig (2010), and equivalence class searches Chickering (2002). The last method searches through Bayesian Network equivalence classes rather than network structures directly. While some of these methods have been shown to scale well to large datasets, the methods that involve optimizing a BIC, likelihood, or posterior probability criteria function will ultimately not scale well to Markov Networks. The criteria function for an undirected model does not factorize in a manner that avoids recomputing the entire score. It is possible to use a Bayesian network rather than a Markov Network, but a Bayesian Network makes strong assumptions about the underlying distribution that are greatly relaxed by Markov Networks. This means Score and Search algorithms are one possible approach to explore; however, this approach is not preferred because of overfitting concerns, the size of the search space, and the fact that it can only learn Bayesian networks. While Bayesian Networks may be powerful in some domains, it is better to pursue approaches that have better guarantees and more representational power, such as Regression Based methods (Section 6.4).

6.3 Constraint Based

Constraint based algorithms focus on learning the graph structure through conditional independence testing. These methods assume that it is possible to recover the distribution's factorization by statistically analyzing the data with standard hypothesis testing methods, such as $\mathcal{X}^2$ tests. Note that hypothesis testing with continuous

random variables is much more challenging and can be intractable in some cases. Since the constraint based approaches are only dependent upon statistical testing, they are better suited for learning Markov networks than the Score and Search algorithms. This benefit is derived from the fact that these methods do not need to compute a global criteria function; however, the lack of a global criteria function can also be viewed as a drawback.

This drawback is best understood by analyzing the simplest constraint based algorithm, SGS Spirtes et al. (1989), that attempts to perform every possible conditional independence test. The SGS algorithm starts with a fully connected graph, and deletes edges that directly connect random variables if those variables are independent. However, two variables are only determined to be independent if they are conditionally independent for all possible random variable subsets that do not include those two variables. This algorithm clearly does not scale to large problems, because the total number of possible conditional independence tests grows combinatorially. While this method will find the true factorization if all statistical tests are sound, it is not possible to apply to real applications. This means that the total number of conditional independence tests needs to be restricted, and without a global criteria all approximate algorithms lose the guarantee of even a local maximum in the general case.

While there are no guarantees in the general case, under certain assumptions most constraint based algorithms perform well and can scale to larger data sets. The Grow and Shrink algorithm (GS) is able to scale to very large data sets by estimating the Markov blanket for each random variable Margaritis and Thrun. (1999). Given the estimated Markov blanket for each random variable, the GS algorithm then recovers a Bayesian Network from the local information. This algorithm's runtime is $O(m^2 + n^3|D|)$, where $|D|$ is size of the training set, n is the number of random variables, and m is the number of edges in the graph. While this algorithm may scale well to a large number of random variables, it will not scale well on E+ data for two reasons: the size of the E+ dataset (millions of data vectors Sanyal et al. (2012)) and

the total number of random variables being modeled (∼300). Ignoring the dataset cardinality issue, the cubic run time results in a very large number of computational steps. Additionally, the number of conditional independence tests required by this algorithm are only polynomial if the Markov blanket for each random variable is bounded. In the worst case the algorithm reverts to an exponential problem, because it will require an exponential number of conditional independence tests.

Another work Pellet and Elisseeff (2008) shows that algorithms that approximate the Markov blanket perform better at extracting casual structures and scale better to large datasets. While we are interested in a method for approximating an undirected graph, approximating the Markov blanket for each random variable easily allows an algorithm to extract the undirected model. In fact this work proves that if an algorithm has the exact Markov blanket for each random variable, then the algorithm will find the true casual model. A casual model is a Bayesian Network in which edges imply causality, which represents a stronger probabilistic relationship. The process requires the algorithm to construct the moral graph for the causal model, where a moral graph is the undirected graphical model that represents the same distribution as the directed model.

While it is computationally intractable to extract the exact Markov blanket for each random variable in large problems, the algorithm uses a different method than GS to determine the Markov blanket. The algorithm uses feature selection algorithms to determine the Markov blanket for each random variable by using backward-selection based linear regression and stepwise selection linear regression. The authors of Pellet and Elisseeff (2008) also explored a backward-selection method combined with SVM regression, called Recursive Feature Elimination (RFE) Guyon et al. (2002), but determined that the method is too computationally expensive even though it allows for the discovery of nonlinear dependencies within the Markov blanket. Feature selection was discussed previously in Chapter 4. In addition, a wide assortment of regression methods are described in Chapter 3. While the proposed algorithm in Pellet and Elisseeff (2008) is more computationally appealing than

the existing Constraint Based and Score and Search methods, it assumes that each variable is Gaussian distributed for use within their feature selection algorithms, making it less general than the other methods. In addition, this dissertation's results illustrate that a genetic algorithm combined with ICOMP(IFIM) is better at feature selection than stepwise selection when using linear regression models. Stepwise selection also generally selects better features than backwards and forwards selection Miller (2002). Therefore, methods that use feature selection to determine the dependent variables are much more computationally feasible. While all components used in Pellet and Elisseeff (2008) have polynomial runtimes (except for the algorithm used to convert a moral graph into a causal model, which is exponential in the worst case), the feature selection methods used are not adequate. However, the idea to use feature selection to determine dependent variables via regression is very intriguing and has lead us to explore other feature selection based methods. These types of approaches are referred to as Regression Based methods and are discussed further in Section 6.4, because they are much more scalable methods than the conventional Score and Search and Constraint Based methods. In addition, the regression based methods presented in this report have a polynomial worst case runtime, while all the other methods are exponential in the worst case.

6.4 Regression Based Method

The regression based structure learning method assumes that it is possible to determine all conditional dependencies among the random variables by assuming that each variable is a functional result from a subset of all random variables. This concept is best presented in Linear Gaussian Bayesian Networks, where each random variable is Gaussian distributed, but each dependent random variable's Gaussian distribution is a linear combination of its parents. Therefore, one can clearly learn the structure for a Linear Gaussian Bayesian Network by performing a series of linear regressions with feature selection.

While the regression based approach is less conventional, it has proven to be extremely scalable. For example, in Gustafsson et al. (2004) regression based methods were used to learn large undirected graphical model structures in Gene Regulatory Networks. Microarray data generally contains thousands of random variables and very few samples. In that particular work the algorithm for building the graph structure is fairly straightforward. If the regression coefficients are non-zero, then there exists a dependency between the response random variable and the random variable associated with each non-zero regression coefficient. While the concept is simplistic, the work used lasso regression to determine the dependencies, which tends to have the ability to weight irrelevant features towards zero. This makes it an ideal approach for determining the graphical structure, because it has built-in feature selection through its regularization term. While this method can learn a general undirected graph structure, it may not be possible to extract an overall joint distribution from the resulting graph. In fact, the work focused on analyzing the overall graphical structure and the distribution over the number of dependencies within the graph, rather than the actual joint distribution represented by the graph.

Another regression based work Dobra et al. (2004) focuses on recovering a joint distribution from the factorized graph. In addition, this method is presented in a general manner and allows for the use of any feature selection method. However, this method assumes that the overall joint distribution is sparse and represented by a Gaussian with $\mathcal{N}(0, \Sigma)$. This type of Markov Network is called a Gaussian Graphical Model (GGM), and it is assumed that the joint probability is represented by the following:

$$\frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} exp(-\frac{1}{2}(x-\mu)^T\Omega(x-\mu)) \tag{6.6}$$

where Σ is the covariance matrix, μ is the mean, and Ω is the inverse covariance matrix or precision matrix. While this assumption is strong, the method assumes that the data is centered prior to application. Based on this assumption and the assumption that the residual error for a variable is also $\mathcal{N}(0, \Psi)$, it appears that

the work assumes application of standard linear regression combined with Bayesian feature selection via a Wishart prior. Nonetheless, this approach learns a Bayesian network over the variables. The Bayesian network is then converted to an GGM by using the regression coefficients and the variance for each regression to extract the precision matrix Ω. The precision matrix is recovered by the following computation:

$$\Omega = (1 - \Gamma)^T \Psi^{-1} (1 - \Gamma) \tag{6.7}$$

where Γ represents an upper triangular matrix with zero diagonals and the non-zero elements represent the regression coefficients, and Ψ^{-1} represents a diagonal matrix containing the variances for each performed regression. There are methods for statistically learning Ω from the data directly, but these methods requires computing Σ^{-1}, the inverse covariance matrix, which is equivalent to Ω. The inverse operation requires an $\mathcal{O}(V^3)$ runtime, where V is the total number of random variables. In addition, it is not guaranteed that Ω will be sparse when estimating Ω directly. Avoiding the computational overhead cost for computing Ω directly is very important when V is very large, which is the case with the E+ random variable set.

In addition, this work illustrates that it is possible to recover the Σ using the precision matrix, but it is not necessary because inference in a GGM can be performed by using the informational form of the distribution:

$$p(x) \propto exp(x^T J x + h^T x) \tag{6.8}$$

where $J = \Omega$ and h is the potential vector; this method assumes that h is a vector of zeros. Based on the work presented in Willsky et al. (2008), it is possible to relax the assumption that h is zero. However, computing h requires Σ by inverting Ω, which is computationally expensive even with the efficient method presented in Dobra et al. (2004).

Overall, this particular regression based approach is fairly robust and computationally feasible for a large number of random variables. However, the method will produce different joint probability models if the variable ordering is changed. Thus, the resulting joint probability distribution is heavily dependent upon the considered variable ordering, because the method assumes that only future variables in the order can be predictors for the current variable under regression analysis. This assumption ensures that the resulting graph structure is well formed and produces a valid Bayesian Network, but requires the method to search the set of all possible orders to determine the best structure. This problem is addressed by scoring the individual variables and greedily ordering the variables in ascending order based on their assigned score.

While Dobra et al. (2004) also presents a method for greedily selecting a good order, Li (2007) builds on this approach and removes the order requirement without adding additional assumptions. The method presented in Li (2007) applies the Wishart prior to the precision matrix, and uses modified Lasso regression to perform feature selection. By shifting the prior to the precision matrix from the individual lasso regression coefficients, it is possible to propagate the prior distribution to the regression coefficients, allowing the modified Lasso regression method to perform Bayesian feature selection. In addition, shifting where the prior is applied allows Li (2007) to prove that regardless of variable order, all possible resulting Bayesian networks encode the same undirected GGM joint probability. This means that it is possible to compute the MAP estimate of Ω, which has very appealing properties such as avoiding overfitting. In addition, the method introduces a way to transform the Lasso regression formulation into an SVM regression problem, where the solution to the SVM regression problem is also the solution to the Lasso regression problem. This transformation allows the method to detect nonlinear dependencies among the random variables.

While the method presented in Li (2007) can efficiently estimate the GGM that governs the joint distribution over the E+ random variables, it is not clear how it will perform on the E+ data set Sanyal et al. (2012), which has many more samples

than variables. The method is intended to work with small sample size data sets that contain a large number of features, such as microarray data. The generated E+ data sets contain 30,540 data vectors per simulation and approximately 300 random variables. This means that traditional methods for solving the Lasso regression method are not able to scale to our problem size. We discuss addressing this issue in Section 5.1.4.

Additionally, adjusting the data set's density could result in an estimated Ω that is not sparse. The Lasso regression to SVM regression transformation requires transforming the initial data matrix before computing the kernel matrix and transforming the response signal as well. The standard linear data matrix X has dimensions $\mathcal{N} \times \mathcal{P}$, with $\mathcal{N}$ samples and $\mathcal{P}$ features. Under this nonlinear method, the data matrix is transformed to X^T, while the response, Y, is transformed to $X^T Y$. The data matrix transformation causes the data matrix to become an $\mathcal{P} \times \mathcal{N}$ matrix. This means that the resulting kernel matrix has dimensions $\mathcal{P} \times \mathcal{P}$. While the kernel matrix will now fit in memory, solving the SVM optimization method with such a dense kernel matrix may produce a regression solution that is not sparse*. This means that the transformed Lasso regression method may lose its ability to select sparse predictor features, which in turn can deteriorate the quality of the estimated Ω. Another thing that must be considered is whether Lasso regression (or the modified version) selects the true predictors. Even though the SVM and Lasso methods, based on the hyper parameter settings, will find the globally optimal regression parameters, there are no true guarantees that the resulting model will be the most parsimonious model. Therefore, we are avoiding Li (2007)'s Lasso to SVM transformation, and only exploring the linear dependencies among the variables, because we can use the E+ approximation models (Chapter 5) to fine tune parameter estimates directly, which will compensate for under represented nonlinear dependencies.

* Sparse in this instance means that most of the predictors have a zero coefficient

Despite the shortcomings with these previous approaches, regression based methods are the most computationally feasible and scalable method currently presented in the literature. Therefore, it is the most ideal approach to tackle approximating E+. However, this approach does require assuming the joint distribution is approximately Gaussian as well as and finding a method that facilitates solving arbitrarily large Lasso regression problems.

6.5 Approach

Given that the regression structure learning method has the most scalability, we only need to address the Lasso regression component's scalability. Using the Lasso regression technique presented in Section 5.1.4, it is possible to solve arbitrarily large Lasso regression problems. This means we are able to use the regression approach to learn a GGM model either using the method outlined in Section 6.5.1 or Section 6.5.2.

6.5.1 Direct GGM Learning

Combining the Gaussian formulation with a scalable Lasso regression method allows us to fit a GGM model directly. The direct method makes zero assumptions about local priors or a global prior, the Bayesian method assumes a global Wishart prior and is discussed in the next section. The learning process directly solves for Γ by using a user defined variable order and solving $N-1$ regression problems, where N is the total number of variables. The variable order specifies a presumed dependency structure, because variables are only dependent upon the ones following them in the order. In more mathematical terms, we solve the following regression problem:

$$x_i = \sum_{j=i+1}^{N} \beta_j x_j + \epsilon_i \tag{6.9}$$

where x_i represents the response variable, or child variable in graphical model terms, x_j represents the predictors, or the potential parent variables, and ϵ_i represents the Gaussian error term, $N(0, \psi_i)$. Under this assumed relationship, we fit a Lasso regression model to the first $N-1$ variables and use the resulting β's to construct the triangular matrix Γ.

After determining β_j, we estimate Ψ_i using the standard unbiased regression variance estimator, which is defined as follows:

$$\psi_i = \frac{\sum_{n=1}^{|D|}(x_{ni} - \sum_{j=i+1}^{N} \beta_j x_{nj})^2}{|D| - 2} \tag{6.10}$$

where $|D|$ represents the total number of samples. We estimate the variance for the first $N-1$ variables in the order, and the final variable's variance, ψ_N, is estimated by fitting a $N(0, \psi_N)$ to variable x_N. Computing the final GGM's precision matrix, Ω, requires computing the inverse variance for all variables, Ψ^{-1}. While the matrix inverse operation is generally expensive, the matrix Ψ is a diagonal matrix, which makes the inverse operation O(N) rather than O(N^3). The Ψ matrix contains each variables' estimated variance along the diagonal.

Once we obtain Γ and Ψ^{-1}, we estimate the precision matrix using Eq 6.7. Having an estimate for Ω allows us to perform inference over the N variables without performing any matrix inverse operations. While avoiding the $O(N^3)$ operation for $\sim$300 variables provides small computational savings, it will provide much more computational savings if we scale to a few thousand variables. In addition, it allows us to perform inference without computing marginals, because we use the unnormalized joint distribution directly, discussed in Section 6.5.3

6.5.2 Bayesian GGM Learning

Unlike the direct method, the Bayesian approach assumes a global Wishart prior for the precision matrix. Using this global prior over the precision matrix allowed

Li (2007) to prove that the Bayesian approach estimates a globally optimal precision matrix over all possible variable orders. That is to say, under the Bayesian formulation presented in Li (2007), all variable orderings should theoretically produce the same precision matrix.

The Wishart prior used in Li (2007) is defined as $W(\delta, T)$. δ represents a user defined hyperparameter and T is a hyperparameter diagonal matrix whose entries are governed by the following distribution:

$$P(\theta_i) = \frac{\gamma}{2} exp(\frac{-\gamma\theta_i}{2}) \tag{6.11}$$

where γ is a user defined hyperparameter. Using the above prior and some additional derivations, Li (2007) derived the following maximum a posteriori (MAP) distributions for all β and all ψ^{-1}:

$$P(\beta_i|\psi_i, D) \propto exp\left(\frac{\sum_{n=1}^{D}(x_{ni} - \sum_{j=i+1}^{N}\beta_{ij}x_{nj})^2 + \sqrt{\gamma\Psi_i}\sum_{j=i+1}^{N}|\beta_{ij}|}{-\psi_i}\right) \tag{6.12}$$

$$P(\psi_i^{-1}|\theta_i, \beta_i, D) \propto P(D|\Psi_i^{-1}, \beta_i, \theta_i)P(\beta_i|\Psi_i^{-1}, \theta_i)P(\psi_i^{-1}|\theta_i) \sim$$
$$Gamma\left(\frac{\delta + 1 + N - 2i + |D|}{2}, \frac{\sum_{j=i+1}^{N}\beta_{ij}^2\theta_i^{-1} + \theta_i^{-1} + \sum_{n=1}^{|D|}(x_{ni} - \sum_{j=i+1}^{N}\beta_j x_{nj})^2}{2}\right) \tag{6.13}$$

Maximizing $P(\beta_i|\psi_i, D)$ with respect to β_i is equivalent to solving a Lasso regression problem with the regularization hyperparameter λ set to $\sqrt{\gamma\psi_i}$ Li (2007). The original authors derived these formulations to work with microarrays, which typically contain 10,000 to 12,000 variables, but have very few samples. This allowed the authors to use conventional optimization methods to solve for β_i. However, the E+ data set used in this work contains several million data vectors, which mostly invalidates conventional

optimization approaches. Rather than using the fast grafting algorithm[†] used by Li (2007), we solve the Lasso regression problems using ADMM (Section 5.1.4).

After obtaining β_i's MAP estimate, it can be used to maximize $P(\psi_i^{-1}|\theta_i, \beta_i, D)$ with respect to ψ_i. However, $P(\psi_i^{-1}|\theta_i, \beta_i, D)$ is dependent upon the hyperparameter θ_i and Li (2007) noted that there is not a known method to analytically integrate out the hyperparameter. This means numerical methods are required to approximate the integral over the hyperparameter. There are many numerical methods for computing approximates to definite integrals, such as Trapezoidal method and Simpson's Rule. However, the integral over θ_i is unbounded from above, because θ_i's values exist in the interval 0 to ∞, which means Eq. 6.13 must be transformed to a bounded integral for the numerical methods to be applicable. Given, Eq 6.13's complex nature and Li's Li (2007) recommendation to use sampling to approximate ψ_i^{-1}, we elected to use $\mathrm{E}[\psi_i^{-1}|\theta_i, \beta_i, D]$ as our estimate for ψ_i^{-1}, which is the MAP estimate under a Gamma distribution:

$$\psi_i^{-1} = \frac{\delta - 1 + N - 2i + |D|}{\sum_{j=i+1}^{N} \beta_{ij}^2 \theta_i^{-1} + \theta_i^{-1} + \sum_{n=1}^{|D|} (x_{ni} - \sum_{j=i+1}^{N} \beta_{ij} x_{nj})^2} \tag{6.14}$$

Given a fixed θ_i^{-1}, we can estimate a ψ_i^{-1} sample by computing the maximum likelihood estimate (MLE) (Eq 6.14). By computing multiple MLE samples according to the θ_i's distribution defined in Eq 6.11, we are able to estimate $\mathrm{E}[\psi_i^{-1}|\theta_i, \beta_i, D]$ using weighted sampling. In order to use weighted sampling, we sample Eq 6.11 using its CDF and a uniform distribution on the interval [0,1]. This means our final ψ_i^{-1} estimate is computed using the following equation:

$$\psi_i^{-1} = \frac{1}{M} \sum_{j=1}^{M} \hat{\psi}_j^{-1} P(\theta_j) \tag{6.15}$$

[†] A gradient based constrained set optimization method.

where M is the total number of samples and $\hat{\psi_i}^{-1}$ is a sample computed using Eq. 6.14. After a few iterations between estimating βs and ψ^{-1}s, we use the final estimates to compute the GGM's precision matrix (Eq. 6.7).

6.5.3 Inference

Given a GGM, we can estimate the E+ building parameters using the GGM's informational form, which is presented in Eq 6.8, to solve the original optimization problem presented in the introduction (Eq 6.1). Using Eq 6.8 to perform inference allows us to avoid computing the normalization component, which requires inverting Ω. Applying the informational form to Eq 6.1 allows us to convert the original probability equation into the following log likelihood equation:

$$\log P(X) = -\sum_{i=1}^{|X|} x_i^T J x_i + h^T x_i \tag{6.16}$$

where x_i represents a complete data vector and h^T represents the GGM's mean, which is assumed to be zero for this work.

In order to perform inference, we simply must fix the observations, i.e., the known observations or evidence, per vector x_i and fill in the building parameter values that maximize Eq 6.16. We may use any optimization method to maximize the log likelihood function with respect to the building parameters. For example, we could use gradient ascent by computing derivatives with respect to the building parameters or hill-climbing with random restart. In this work, we elected to use a genetic algorithm to optimize the building parameters. Specific genetic algorithm details are discussed at length in Section 4.2. We set the GA's population size to 200, max generations to 10, and used single point cross-over. Mutation was not allowed.

However, we used a gradient optimization method on our MO2 experiments described in Section 6.6. Our results show that the GA is adequate for a small parameter space inference, but the gradient methods are best for a larger parameter

space. Our discussion section provides a detail analysis between the two optimization methods (Section 6.8).

6.6 Experiments

In order to initially assess how well the GGM estimates building parameters, we sampled 10 building simulations for fitting the models. These simulations were sampled from the Markov Order 1 (MO1) residential building simulation data set. The MO1 simulation data set contains 299 simulations. Each simulation has 90 outputs, 156 building parameters, an operation schedule, and a weather file. Two simulations were run per building parameter. One simulation with a building parameter was set to its maximum value and the other with the same variable set to its minimum value. All other building parameters remained at their average value.

Using these simulations, we built two GGM models. The first GGM was built using the Direct method described in Section 6.5.1 and the other was built using the Bayesian method described in Section 6.5.2. Both GGM models were tested using 50 simulations randomly sampled from the remaining 289 MO1 simulations. These test simulations have ground truth building parameters associated with their simulation output, allowing us to directly estimate how closely the models approximate the known answer.

In addition, we computed a random guess for each test simulation using a uniform distribution as well. The random solutions provide a baseline performance that can be compared against the GGMs. While it may seem uninformative to compare against uniform random guessing, a vast majority of the MO1 parameter buildings are set to their average value, which maps to 0.5 when normalized using a min/max scaling approach. This means the random guessing method should be fairly competitive because the expected value for a uniform distribution between 0 and 1 is 0.5.

We also built a GGM using 250 Fine Grain (FG) simulations sampled at random without replacement. The FG data set contains building simulations built using a

brute force sweep over 14 building parameters. The FG GGM was evaluated using 150 simulations sampled from the FG data, after the training simulations were removed.

In addition to these two experiment sets, we built a GGM using the full MO1 data set. Using this model, we tested parameter inference on 300 sampled MO2 simulations. These simulations are the same ones that we used in Chapter 5's experiments.

6.7 Results

In order to enforce any building parameter value constraints, we standardized all data using the allowed or expected minimum and maximum values. This means the transformed minimum value now corresponds to 0 and the transformed maximum value now corresponds to 1. After scaling the data, the means for each variable are estimated independently and subtracted from all samples. This step is required to comply with the model's assumption that the data is normally distributed according to $N(0, \Sigma)$. In addition, we used the absolute error difference between the predicted value and the exact building parameter value. Given that all values are between 0 and 1, a squared error metric will produce overly optimistic performance estimates by shrinking the error through the squaring operation. With this information in mind, the Bayesian and Direct error figures are scaled between 0 and 1, which means values closer to 1 represent poor performance and values closer to 0 represent good performance. In addition, the Direct vs Randon and Bayesian vs Random are between -1 and 1 due to the scaling.

The parameter inference results are organized as follows: Section 6.7.1 presents inference results using the model built from 10 MO1 simulations; Section 6.7.2 presents inference results using the model built from 250 FG simulations; and Section 6.7.3 presents results from testing the GGM built with the MO1 data set on 300 simulations sampled from MO2 data set. Note, the method comparison figures in Section 6.7.1 are read as follows: values closer to -1 indicate the first method performs

better than the second (i.e., Direct > Random), and values closer to 1 indicates the second method performs better than the first (i.e., Random > Direct).

6.7.1 MO1 Results

Analyzing Figure 6.1 indicates that the learned Bayesian GGM is not able to infer the building parameters for 9 variables – 3, 12, 19, 31, 38, 49, 62, 75, and 81. The inability to properly infer these variables is either due to the small training set, the model's underlying Gaussian assumptions being invalid for these variables, or the model not adequately capturing the probabilistic dependencies for these variables. The first problem is a possibility, but is the least likely given the smooth variance on all predictions. A key component within a Bayesian approach is its ability to smooth prediction variability by incorporating priors. Comparing the error variance in Figures 6.1 and 6.2 indicates the priors are indeed reducing the overall error variance by producing more consistent estimates. However a consistent estimate does not necessarily mean a more accurate estimate; it just implies predictions will concentrate within a particular area due to the priors.

This leaves the other two options as the most likely reasons. However, we have not yet verified which reason is leading to the poor estimation. Either the variables have nonlinear dependencies, which are not possible to detect using a Lasso regression structure learning approach, due to the model's linear nature, or the true distribution for these variables is not Gaussian. We could test whether the individual variables are Gaussian distributed, but we need to actually verify whether or not the conditional distributions for each individual variable are Gaussian distributed, which is much harder to determine.

In addition, variables 113, 115, 117, and 120 have poor average performance and high variance, which indicates the probabilistic relationship is only partially represented for the variables. While slightly speculative, this is most likely due to the

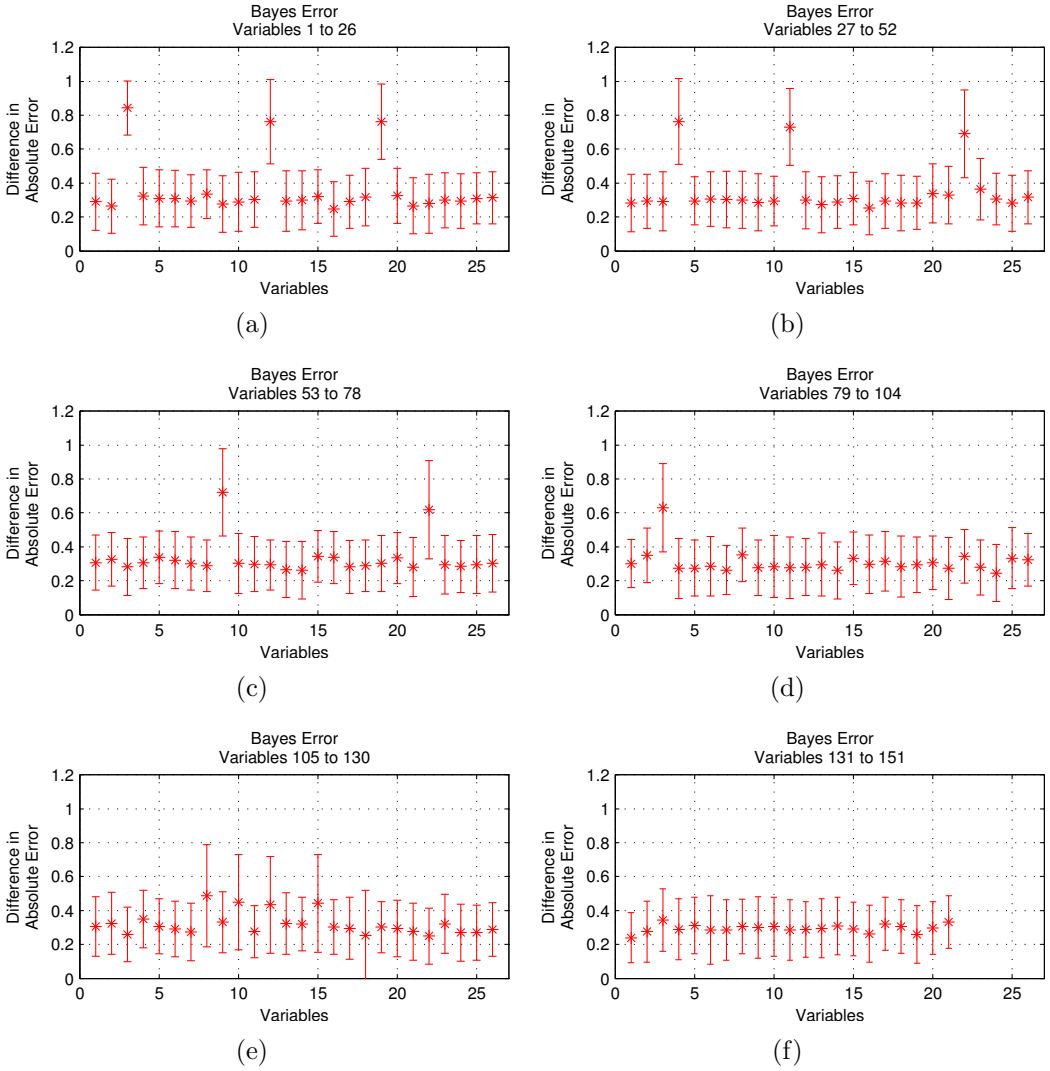


Figure 6.1: Bayesian GGM's error on 50 randomly sampled MO1 simulations.

small number of training simulations[‡]. The Bayesian GGM provides better estimates than the Direct GGM estimates for these building parameters (Figures 6.1 and 6.2), which indicates the priors could be correcting for insufficient simulation samples. Alternatively, these variables have linear and nonlinearnonlinearnonlinearnonlinear dependencies, and the priors are correcting for the bias induced by the GGM model only containing the linear dependencies.

Analyzing Figure 6.2 indicates that the GGM learned using the Direct method is not able to estimate the building parameters for 14 variables – 3, 12, 19, 31, 38, 49, 62, 75, 81, 113, 115, 117, 120, and 123. However, despite the model's higher error variance, the Direct GGM produces better estimates for variables 3, 12, 19, 31, 38, 49, 62, 75, and 81 than the Bayesian GGM.

The most important conclusion is gained by analyzing Figures 6.1 and 6.2 together, which shows that the Bayesian GGM and Direct GGM produce equivalent predictions on average for a vast majority of the variables. However, the Bayesian GGM produces more consistent estimates. This means the Bayesian GGM will either produce consistently accurate estimations or consistently inaccurate estimations, rather than producing highly variable estimations when the underlying structure is incorrect, unlike the Direct GGM. However, the Bayesian GGM's learning process is much more computationally expensive, because it makes multiple iterations across the entire data set. These multiple passes allow the method to refine the Lasso regression regularization parameters for each variable. When all assumptions are met, this leads to a more accurate GGM.

Figure 6.3 presents results from comparing the Bayesian GGM against randomly guessing using a uniform distribution. The results indicate that the uniform distribution is able to randomly produce better estimates for most variables between 1 and 26, 53 and 78, 79 and 104, and between 105 and 130. However, there are also several variables that randomly guessing performs extremely poor on as well.

[‡] We used 10 E+ simulations as training data, which contains 350,400 data vectors. Each data vector has 300 variables.

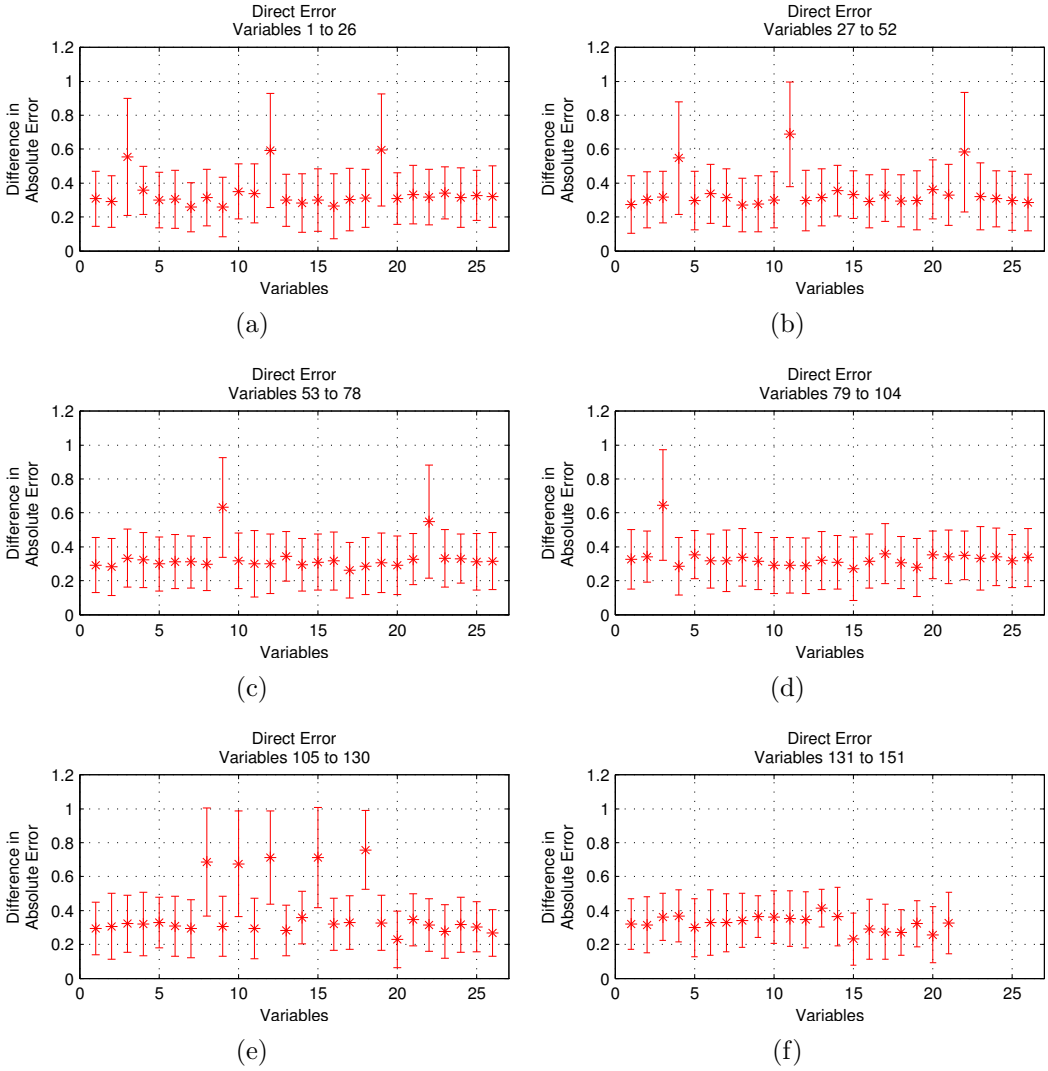


Figure 6.2: Direct GGM's error on 50 randomly sampled MO1 simulations.

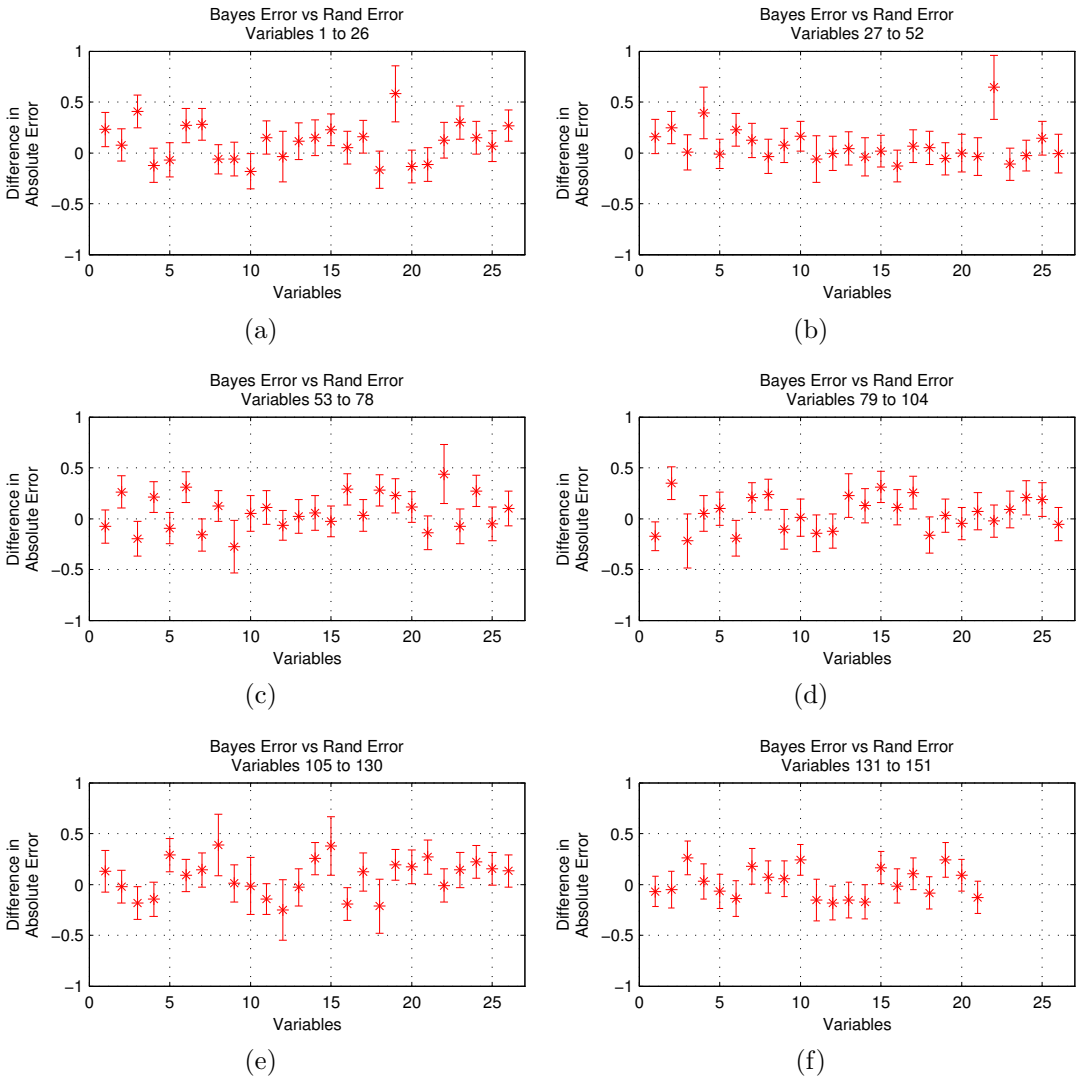


Figure 6.3: Shows Bayesian GGM's error compared against the error from randomly estimating building parameters using a uniform distribution.

This means the results are fairly mixed on this particular data set. In order to fully demonstrate the GGM's capabilities, we need to experiment on data sets whose values mostly map to a uniform distributions expected value.

Figure 6.4 illustrates similar mixed results for comparing the Direct GGM against the uniform distribution. Although, the similar results are expected given the similar performance shown between the Bayesian GGM and Direct GGM models (Figures 6.1 and 6.2). In addition, the difference between the Direct GGM and the uniform distribution has higher variance than the difference between the Bayesian GGM and the uniform distribution. This further confirms that the Direct GGM's error has higher variance than the Bayesian GGM's error.

6.7.2 FG Results

While the MO1 results are mostly inconclusive, they do illustrate that the Bayesian GGM model produces less variable estimates. Therefore, we only experimented with the Bayesian GGM with the FG data, which has resulted in much more definitive evidence that the GGM is performing better than randomly guessing. Analyzing Figure 6.5 illustrates that the building parameter estimates using the GGM are mostly better than the estimates generated by randomly guessing using a $[0, 1]$ random distribution. The overall GGM's absolute error rate is statistically better with 95% confidence than the uniform distribution's error rate – 4.05 ± 0.98 vs 5.07 ± 1.01. However, only the error rates for variables 1, 2, 3, 8, 10, 11, 12, 13, and 14 are statistically better than their random guessing counterpart. The other variables have a lower mean but are not statistically different with high enough confidence.

While the other variables are not statistically different, there is not a clear indication that the GGM model fails to represent these variables. Unlike the results in Figure 6.1, the under performing variables are not grossly inaccurate. This indicates that the GGM overall is successfully modeling the FG building parameters. In fact analyzing Figures 6.5(c) indicates that the GGM isolates the building parameter's

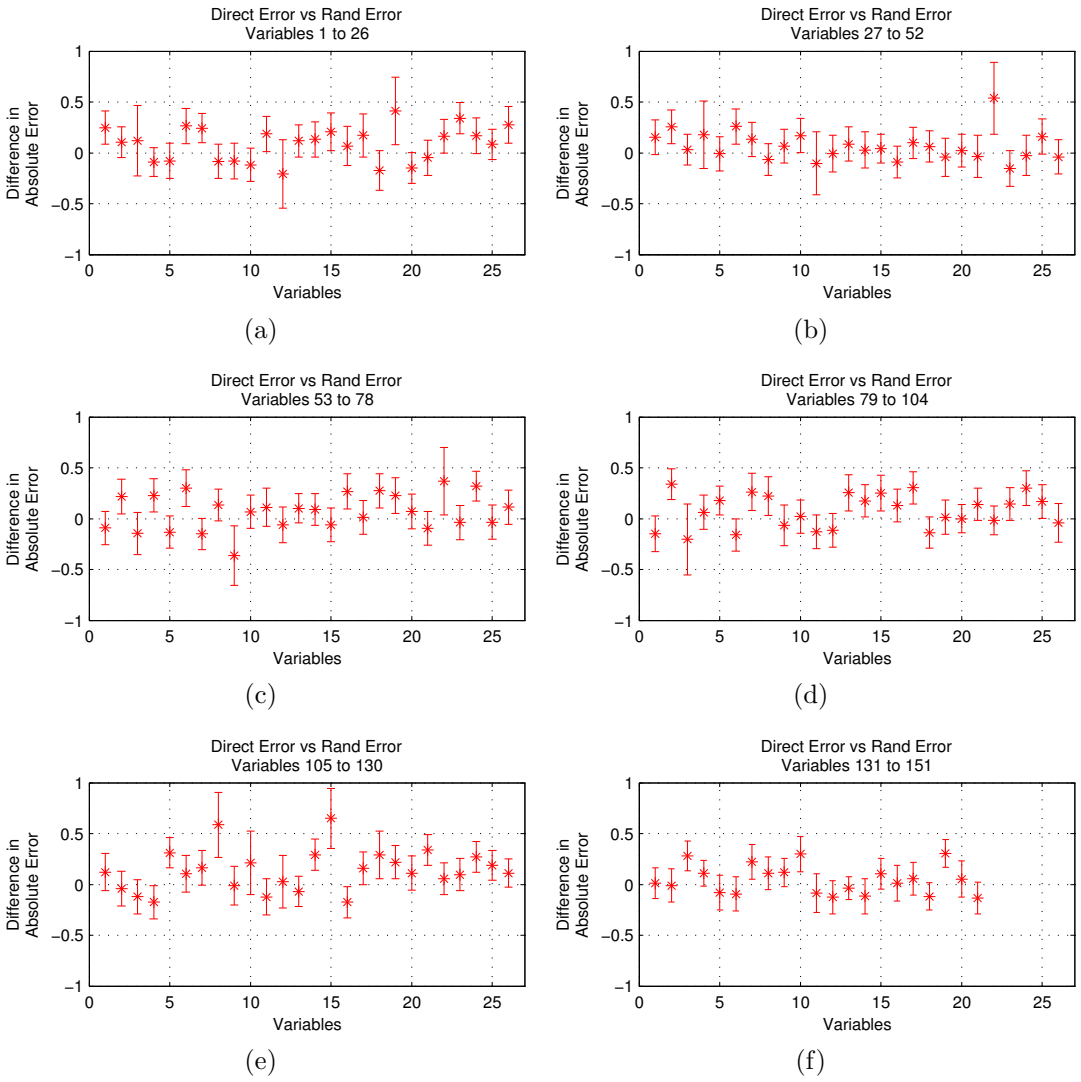


Figure 6.4: Shows Direct GGM's error compared against the error from randomly estimating building parameters using a uniform distribution.

means very well. While the uniform distribution matches the mean and variances (Figure 6.5(d)), it appears the matching is only possible because the actual building parameters have a large distribution range, which mostly centers at 0.5, the uniform distribution's expected value.

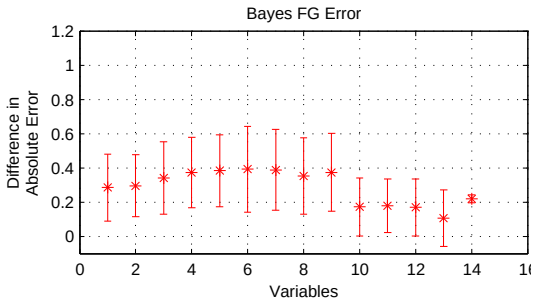
6.7.3 MO2 Results

Similar to the FG data set results, the Bayesian GGM presents excellent performance (Figure 6.6). In the previous two result sections, we compared the overall error average per variable between these two methods. However unlike the MO1 results, these results are very definitive and do not require further elaboration with additional figures. The Bayesian method is better at estimating all building parameters than a uniform distribution, which is clearly seen by comparing Figure 6.6 and 6.7.

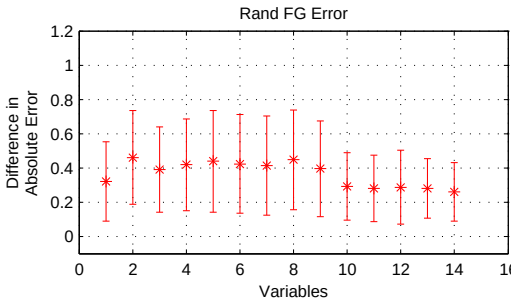
The Bayesian GGM has statistically better overall error — 13.01±0.85 vs 41.6936±2.13. In addition, the model produces statistically better predictions for all variables. While we did not build a GGM using the direct method on this data, we are fairly confident that it will not perform better than the Bayesian model, because it achieves worse performance on the small scale MO1 experiment.

6.8 Discussion

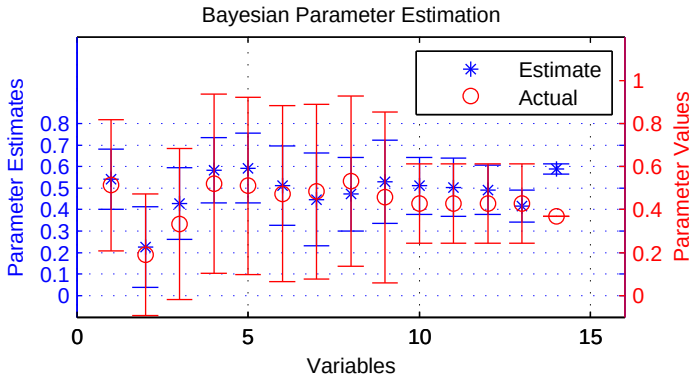
The MO1 versus MO2 experimental results leave a single unanswered question — why are the MO2 results superior to the MO1 results? We believe the answer is actually fairly straightforward. The MO1 and MO2 data sets have building parameters that tend to be very close to their mean values through out all simulations. This means there is a very precise value or target for each building parameter, which is indicated by the actual building parameter variance seen in Figure 6.6. This is why the uniform distribution performed very well on average, but with high variance, on the building parameters that centered at 0.5. Given such a narrow parameter range, a genetic



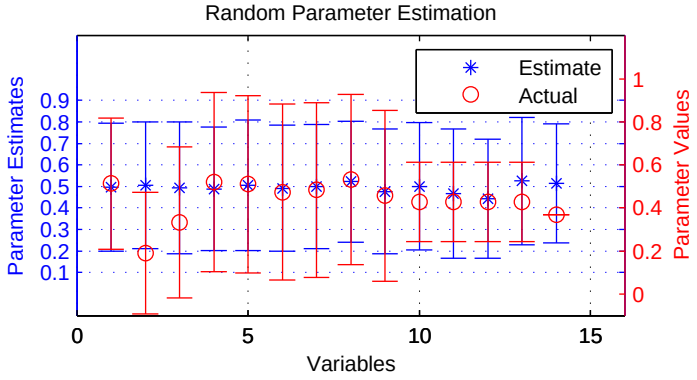
(a) Bayesian



(b) Random



(c) Bayesian Estimates



(d) Random Estimates

Figure 6.5: This figure compares Bayesian GGM's error against a $[0, 1]$ uniform distribution's error on estimating FG building parameters. In addition, it illustrates how the two estimates align with the actual building parameter values.

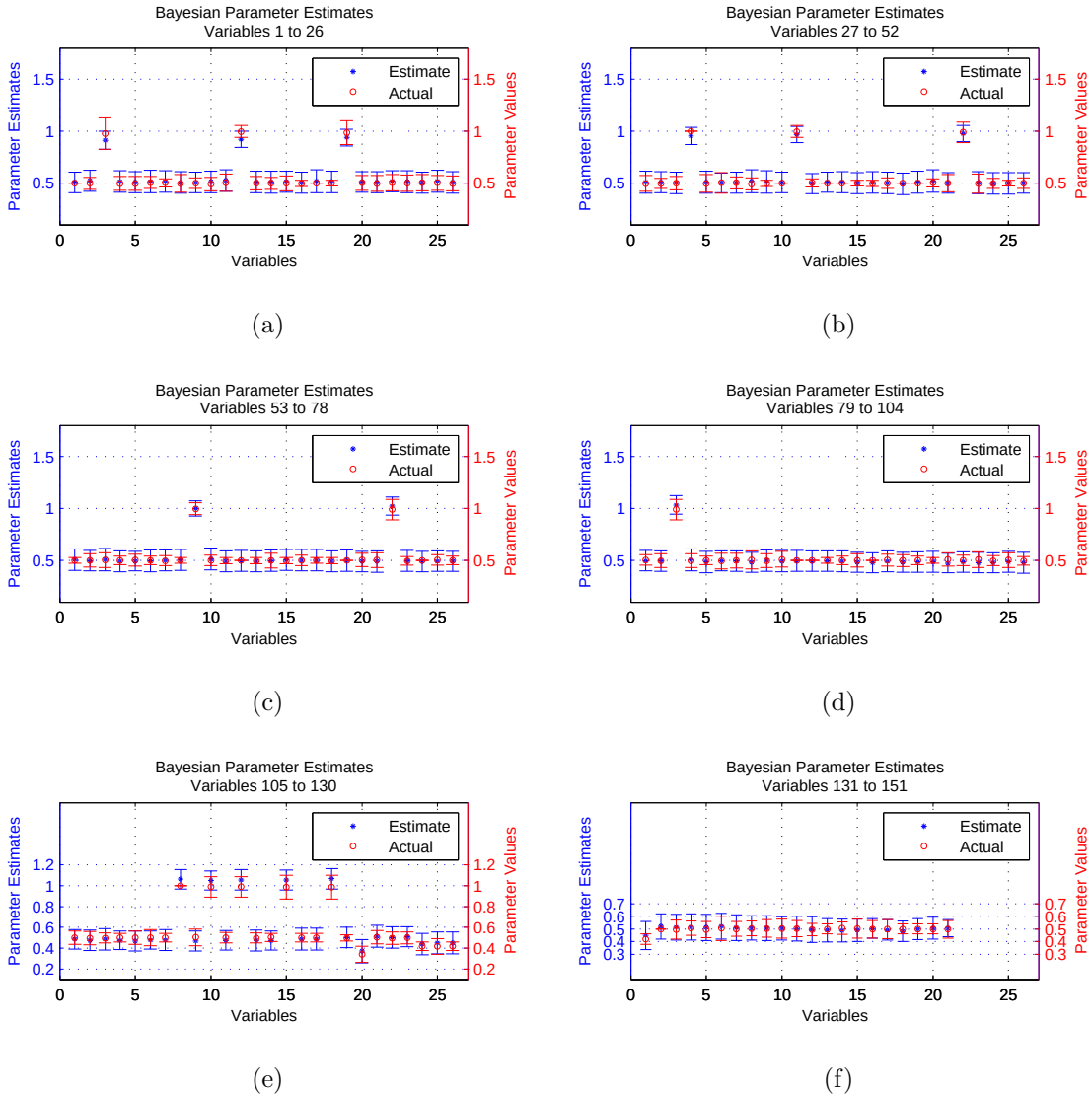


Figure 6.6: Bayesian GGM's parameter estimates compared against the actual parameter values on 300 randomly sampled MO2 simulations.

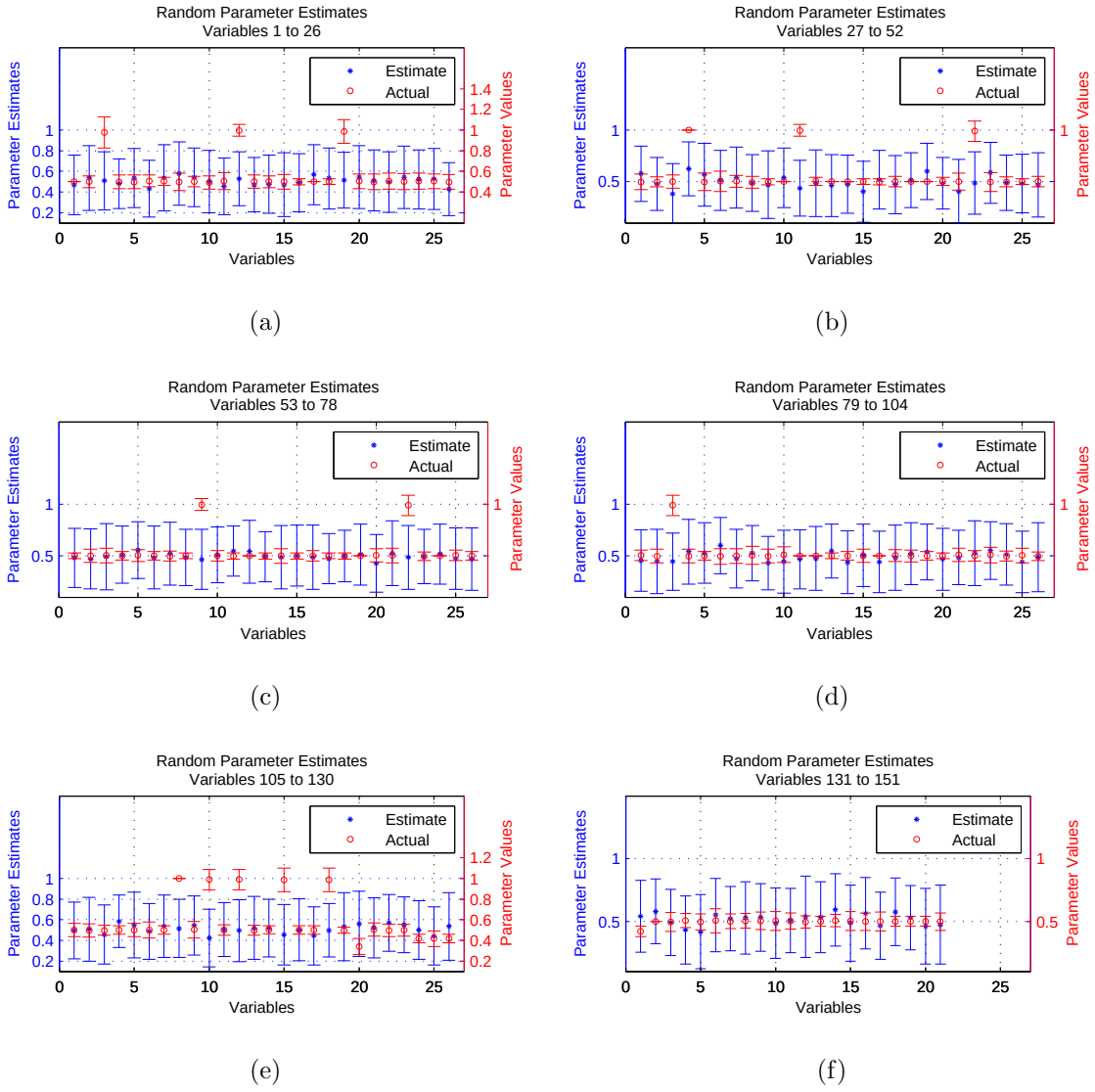


Figure 6.7: Random parameter estimates sampled from a $[0, 1]$ uniform distribution compared against the actual parameter values on 300 randomly sampled MO2 simulations.

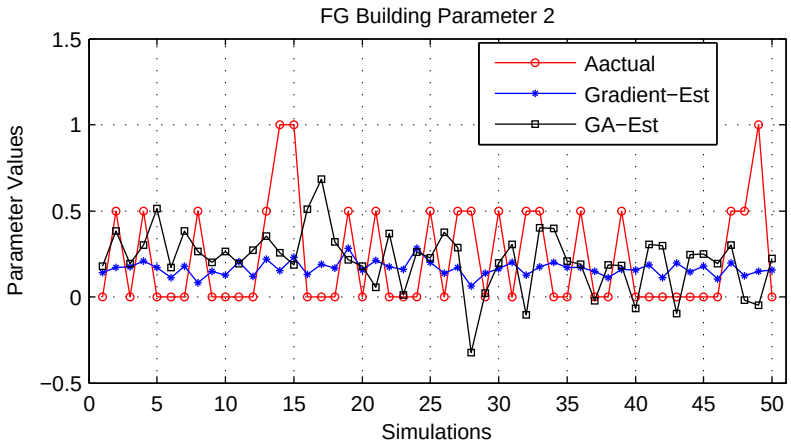


Figure 6.8: This figure compares parameter values estimated using a Genetic Algorithm and a standard gradient optimization algorithm.

algorithm will maximize the likelihood function, but not as finely as a gradient approach. However, on a small enough parameter space, the genetic algorithm is able to achieve excellent performance as seen in Figure 6.5. This implies that as the overall parameter space increases, a slower gradient method is much more advantageous for estimating the building parameters.

In fact, Figure 6.8 compares the gradient and GA estimates from the FG data set on building parameter two. The gradient method produces estimates that are much closer to the building parameter's mean value. In addition, the estimates have much less variability than the GA estimates. However, the gradient's error rate, 0.25 ± 0.17, and the GA's error rate, 0.31 ± 0.22, are only statistically different with 85% confidence. While the confidence is not high enough (95%), this comparison suggests that large parameter estimation problems will be much less accurate when using a GA, because the overall variance in estimation will be magnified as the overall number of building parameters increases. However, this does not imply that a GA can not produce fast parameter estimates, which may be very beneficial in some applications, but it does imply that it is best to use a gradient method for larger parameter estimation problems.

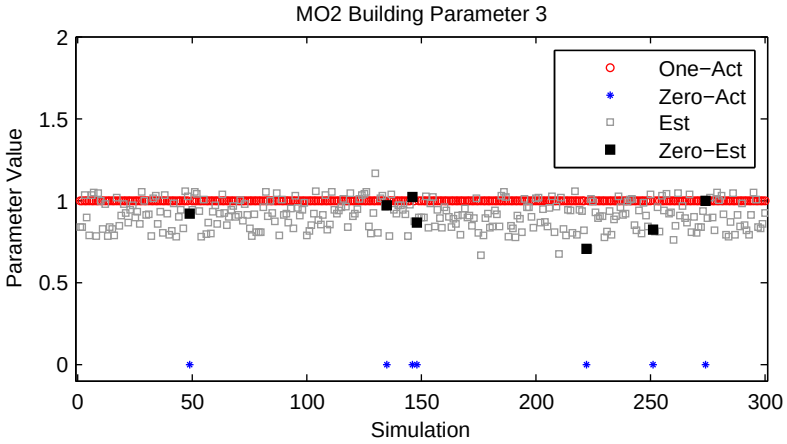


Figure 6.9: This figure highlights that building parameter three has values that occasionally differ greatly from the parameter's mean. In addition, it presents how the Bayesian GGM's estimates correlate with these large deviations.

While the parameters in MO1 and MO2 have well defined means, they have instances where they significantly deviate from their estimated means, which is not well implied by their variances in Figure 6.6. Figure 6.9 illustrates that variable three from Figure 6.6(a) is occasionally zero, which is vastly different from its estimated mean from the MO1 data set — 0.996. In fact, under a standard Gaussian model these variable changes are essentially not representable, because these values are on the distribution's tail. However, it is not impossible for the model to estimate a parameter value towards the distribution's tail, if the observed evidence supports that hypothesis. This implies that variables whose estimates do not have significant shifts towards the tail, either have very little effect on the overall simulation's output as a whole or the model is does not represent the necessary dependencies to represent the shift.

Figure 6.9 shows how the GGM estimates correspond with variable three overall, as well as how they correspond with the building parameter being zero. When the actual building parameter is zero, the simulation between 200 and 250's parameter estimate may be shifting towards the distribution's tail, but the other estimates are

most definitely not shifting. In addition to variable three, other variables within the MO1 data set and the MO2 data set present the same behavior.

However, the FG data set presents similar parameter estimation issues as well. Figure 6.8 illustrates that parameter values which deviate from the mean are harder to estimate as well. The GGM focuses primarily on predicting the parameter's mean, which is expected. A Gaussian model should focus its estimates around the mean, and have difficulty estimating outlier or distance values, because their likelihood's are fairly low. This implies that our models, under their current hyperparameter values, are fitting the means very closely and not allowing the model to explore other possible assignments using a gradient inference method. Using different hyperparameter settings may allow the model to introduce additional variance within the overall estimation process, which may be desirable.

6.9 Results Summary

We have demonstrated that a GGM is able to infer building parameters for the MO1 data set. In addition, we illustrate that a GGM learned using the Bayesian method is able to produce more consistent estimates than the GGM learned using the Direct method. While consistent estimates does not necessarily imply better estimates, the Bayesian method overall produces a better GGM than the Direct method. However, we have not fully illustrated the GGM's capabilities using the MO1 data set. Given that most variables in the MO1 data set, when min/max scaled, map directly to a $[0, 1]$ uniform distribution's expected value, it appears that randomly sampling building parameters is fairly competitive with the GGM's inferred parameters.

However, our MO2 experiments indicate that the uniform distribution is not competitive with building parameters estimates when using a gradient optimization method. This implies that our MO1 results have reduced quality from the variance introduced by the GA optimization method. In addition, we compare the GA and gradient optimization results in Figure 6.8 using the FG data set results. This

comparison confirms that the GA introduces more variance within the parameter estimation process, and leads to our conclusion that it is best to use the slower gradient method on large scale parameter estimation problems.

Additionally, our MO2 and FG experimental results indicated that the GGM performs well at estimating building parameters. Overall, the Bayesian models built using the FG and full MO1 data sets are statistically better than the uniform distribution, which is expected. However, our current GGMs have difficulty estimating parameters that deviate significantly from the mean. This implies we need to explore different hyperparameter settings, which may induce more estimation variance, or possibly a mixture model approach, which will allow more variable means.

6.10 Computer Science Contribution Summary

Learning and fitting distributions via structure identification, i.e. structure learning, is a common problem within the Computer Science community. The paradigm applies to web link analysis, human activity recognition, and many more areas. This dissertation demonstrates using structure learning to fit a sparse GGM to E+ simulation data via a general Bayesian method introduced by Li (2007). While the original Bayesian method will scale to an arbitrarily large number of variables, it will not scale to arbitrarily large data sets, i.e., data sets that contain millions of data vectors. The Bayesian method is dependent upon being able to solve multiple Lasso regression problems, which become increasingly difficult as the total number of examples increase. This dissertation addressed this issue by solving the Lasso regression problems using ADMM (5.1.4), which allows the algorithm to truly scale to large data sets. Essentially, it is now possible to use this approach to identify variable structure and fit a corresponding GGM to any data set.

Conclusion

This dissertation proposes the following automatic simulation calibration process: 1) Identify a model that accurately estimates the real world simulation calibration target from measured sensor data (Chapter 3); 2) Identify the key real world measurements that best estimate the simulation calibration target (Chapter 4)); 3) Construct a mapping from the most useful real world measurements to actual simulation outputs; 4) Build fast and effective simulation approximation models that predict simulation output using simulation input[§] (Chapter 5); 5) Build a relational model that captures inter-variable dependencies between simulation inputs and outputs (Chapter 6); and finally 6) Use the relational model to estimate the simulation input variables from the mapped sensor data, and use either the simulation model or approximate simulation model to fine tune input simulation parameter estimates towards the calibration system.

Step 1

This dissertation has introduced and demonstrated five out the six components outlined above using the building energy simulation domain as the testing and validation area. Given sensor data collected from three residential homes, step 1 is addressed by determining which machine learning technique performed best at predicting whole building energy consumption for the next hour. The experimental

[§] Only required if the overall simulation engine is extremely slow, making it difficult to run many simulations

results show that LS-SVM is the best technique for modeling each residential home. In addition, the results show that the previously accepted method, FFNNs, performs worse than the newer techniques explored in this work: HME-FFNN, LS-SVM, and FCM-FFNN. Lastly, these results show that SVR and LSSVM perform almost equally with respect to CV and MAPE. However, experiments with SVR present poor MBE results, which makes LS-SVM the preferred technique. This work was published in Energy and Buildings Edwards et al. (2012).

In addition, these methods were validated by producing comparable results on the Great Energy Prediction Shootout data set. These validation results are consistent with the existing literature in concluding that FFNN performs best on the original competition data set, and that other types of Neural Networks might perform even better. In addition, these results show that the LS-SVM is the worst performing technique for the Shootout data set, and that shuffling the data improves its performance.

Step 2

Step 2 is addressed by determining which sensors are most important for predicting whole building energy consumption for the next hour. The results demonstrate that a Genetic Algorithm with the ICOMP(IFIM) multi-objective criteria function is able to reduce model complexity, while still giving a reasonable goodness-of-fit. Additionally, these results illustrated that the Stepwise Selection method is sometimes capable of producing smaller sensor subsets than the Genetic Algorithm approach, but the Stepwise Selection models are rarely less complex than the models generated by the Genetic Algorithm, even when the Genetic Algorithm includes additional sensors within the model. In addition, this research introduces a method for ranking the sensors by combining all best models found from the Wrapper techniques, which are able to produce the best models for House 1, House 3, and across all houses. Additionally, using the ranking techniques and Wrapper methods, this work illustrates

some of the effects missing values have on the algorithms. Stepwise Selection performs better when all missing values are set to zero, and the Genetic Algorithm method is fairly indifferent to the missing data approaches. However, it finds its best results generally when missing values are set to zero. Lastly, the Genetic Algorithm with ICOMP(IFIM) and the voting wrapper selection results are compared against the best possible subsets up to size four, which shows that it is computationally infeasible to directly compute a large enough subset that approximates the true best subset. Therefore, the Genetic Algorithm method is the ideal approach for sensor subset selection.

Step 4

While step 4 is optional, the application domain's simulation engine is slow enough to require approximation. Using FFNN and Lasso regression with ADMM, the optional step is addressed by producing E+ approximation models for a residential building. The models use building envelope parameters selected by domain experts, an operation schedule, and weather data. These models are able to successfully predict a majority of the domain expert selected output variables. In addition, this research identifies which output variables require a nonlinear model, based on comparing the FFNN and Lasso models directly. However, these models only have moderate success at predicting sensible heating and cooling loads, and are unsuccessful at predicting the latent cooling and heating loads.

In an effort to improve the E+ approximation load predictions, we incorporate HVAC operating heating and cooling features, which indicate the on and off states for these respective operating conditions. These new features presented mixed results. Some load predictions are improved, while others are unchanged or diminished. Based on these results and Lasso regression's ability to automatically select relevant inputs, we conclude that either better use of existing information or additional information is necessary to better predict the latent load variables.

The Lasso model is able to predict an entire yearly simulation in ∼3 seconds, while the FFNN models can achieve the same execution time when run in parallel. These runtimes are considerably faster than the average E+ runtime (∼2-3 minutes). This performance increase provides improvement to the overall building calibration process, when using the well predicted variables in the tuning objective.

Lastly, the three data sets (Fine Grain, Markov Order 1 and Markov Order 2) allow us to determine that the best E+ approximation model requires multiple models, as discussed in Section 5.4, which can tailor the learning to individual cluster attributes.

Step 5

Step 5 is addressed by adapting the Direct and Bayesian regression based structure learning techniques for Gaussian Graphical Models (GGM) to work with arbitrarily large data sets by leveraging Lasso regression with ADMM. Since both methods are centered around using Lasso regression to determine inter-variable dependencies, naturally extending the regression solver to this scale facilitates general purpose linear dependency structure learning.

Step 6

This dissertation demonstrates that a GGM is able to infer building parameters for the MO1, FG, and MO2 data sets. In addition, it illustrates that a GGM learned using the Bayesian method is able to produce more consistent estimates than the GGM learned using the Direct method. While consistent estimates do not necessarily imply better estimates, the Bayesian method overall produces a better GGM than the Direct method. However, the MO1 results do not fully illustrate the GGM's capabilities, because a uniform distribution is fairly competitive with the model's estimates. Additional exploration indicated that the poor performance is primarily attributed to the GA optimization method. We demonstrated that the GA method

introduces additional variance, and that a gradient method produces estimates that are more consistent with the parameter's mean.

However, this dissertation illustrates that the GGM performs better than randomly guessing on the FG and MO2 data set. The GGM model produces less variance and better overall parameter estimates. In addition, the uniform distribution would be less affective on all data sets if we shifted its range to $[-1, 1]$ and shifted the target parameters to have zero mean, which is required by the GGM model.

Contribution Summary

The following list summarizes all Building Spaces contributions:

- Best predictor for hourly residential electrical consumption (Chapter 3)
- Best sensors for predicting electrical consumption (Chapter 4)
- First general purpose large-scale E+ residential approximation (Chapter 5)

The following list summarizes all Computer Science contributions:

- A novel feature selection method, which uses the estimated ICOMP distribution over the features to select the best ones via voting (Chapter 4)
- Adapting Bayesian and Direct regression structure learning to large-scale datasets, via Alternating Direction Method of Multipliers (Chapter 6)
- General large-scale automated computer simulation calibration process (Figure 1)

Looking Forward

While we have demonstrated the relational model's ability to estimate building parameters estimates, we have only partially demonstrated Step 6. Complete

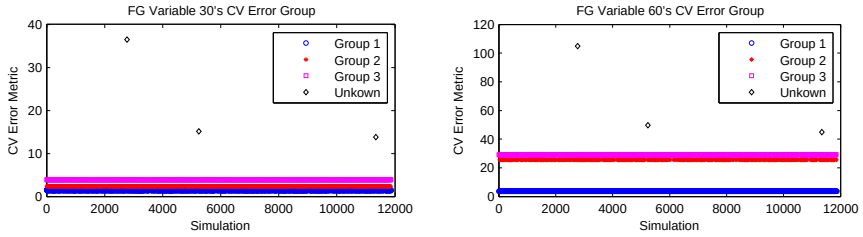


Figure 6.10: Identifies three potential outlier simulations within the Fine Grain data set. The outliers in each figure represent the same simulation across the two figures, i.e. there are three outlier simulations and not six outliers.

demonstration requires estimating building parameters using the available WC1 sensor data and using the estimated building parameters to run E+ simulations. Using our work, ORNL researchers can produce building parameter estimates and repeatably build different GGM models by adjusting hyperparameters or changing the variables used within the model. In addition, they can run the corresponding E+ simulations, and compare the simulation output against the actual building's measurements as well as the building engineer's manually calibrated simulations. These comparisons will provide estimates for how much human time is saved using the automated calibration process. The building engineer model required approximately two months calibration time, which provides the base line for computing all savings with respect to overall building electrical consumption.

Future Directions

Given the very rich E+ data sets and the available computing power to generate millions of simulations, there are many directions in which this work may continue. One possible direction focuses on improving the surrogate's overall estimation accuracy, by using domain experts to identify and isolate internal E+ variables to improve prediction accuracy, which will likely require running additional simulations.

In addition, care must be given to ensure the predictive models are still applicable to other residential buildings.

A second surrogate improvement direction, but a parallel option to the first, is continuing to explore features synthesized from existing data. Using the existing simulation information, we can build new features or approximations based on expert selected features. This direction does not require running additional simulations, but would require additional expert time and computational cost for synthesizing general features.

Alternatively, exploring the observed E+ clustering may potentially lead to models for E+ simulation anomaly detection. Using the E+ clustering property, it may be possible to identify outlier simulations or simulation subroutines. These outlier simulations could either be other undiscovered clusters, actual simulation anomalies, or bugs within the E+ simulation software. Figure 6.10 illustrates an example. In this figure, there are three outlier simulations, which were detected in the Fine Grain dataset.

Comparing the full calibration system against the partial calibration system used for E+ will also be a fruitful study. In this work, domain experts select the most important sensors and specify their mappings to the simulation outputs. While the mapping specification may be required, it would be interesting to see how the calibration process performs using the sensors selected by our voting feature selection method and the sensors selected directly by ICOMP. However, speculations about the result is not possible until estimates for the WC1 building parameters and their corresponding building simulations are available.

Finally, exploring methods for automatically mapping sensor data to simulation output is definitely an interesting area to explore. This problem is especially difficult, because the sensor data is only statistically dependent upon the true building parameters, if and only if the simulation is physically accurate as well. This means exploring models that represent how building parameters influence sensor data, which will ultimately lead to a model that estimates building parameters directly from sensor

data. However, collecting enough data will be non-trivial. The simulation relational model requires generating simulation data using computing power, while correlating sensor data and building parameters requires collecting multiple data sets from many different buildings over time. In addition, it requires a model that is able to handle the sparse under-sampled parameter space. In addition, unlike the simulation data, real world sensor data will always contain missing measurements, which makes relational learning even more difficult.

Bibliography

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In *Proceedings of the Second International Symposium on Information Theory*, pages 267–281. B.N. Petrov and F. Caski, eds., Akademiai Kiado, Budapest, Hungary. 54, 109

American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc. (2009). *2009 ASHRAE Handbook - Fundamentals*. D&R International, Ltd. 25

Bernardo, J., Bayarri, M., Berger, J., Dawid, A., Heckerman, D., Smith, A., West, M., et al. (2003). Hierarchical bayesian models for applications in information retrieval. In *Bayesian Statistics 7: Proceedings of the Seventh Valencia International Meeting*, page 25. 141

Blei, D., Ng, A., and Jordan, M. (2003). Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022. 141

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122. 14, 107, 110, 111, 112, 113

Bozdogan, H. (1987). Model selection and Akaike's information criterion (AIC): The general theory and its analytical extensions. *Psychometrika*, 52(3):345–370. 55

Bozdogan, H. (2003). Intelligent statistical data mining with information complexity and genetic algorithms. *Proceeding of JISS 2003, Lisbonne*, 2:15–56. 56, 58, 104

Bozdogan, H. and Haughton, D. (1998). Informational complexity criteria for regression models. *Computational Statistics & Data Analysis*, 28(1):51–76. 55, 104, 109

Cawley, G. and Talbot, N. (2010). On over-fitting in model selection and subsequent selection bias in performance evaluation. *The Journal of Machine Learning Research*, 11:2079–2107. 114

Chang, C.-C. and Lin, C.-J. (2011a). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm. 32, 47

Chang, C.-C. and Lin, C.-J. (2011b). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm. 109

Chickering, D. (2002). Learning equivalence classes of bayesian-network structures. *The Journal of Machine Learning Research*, 2:445–498. 145

Christian, J., Gehl, A., Boudreaux, P., New, J., and Dockery, R. (2010). Tennessee Valley Authoritys Campbell Creek Energy Efficient Homes Project: 2010 First Year Performance Report July 1, 2009 August 31, 2010. pages 1–126. http://info.ornl.gov/sites/publications/files/Pub26374.pdf. 20

Collobert, R. and Bengio, S. (2001). SVMTorch: support vector machines for large-scale regression problems. *J. Mach. Learn. Res.*, 1:143–160. 109

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38. 36

Dobra, A., Hans, C., Jones, B., Nevins, J., Yao, G., and West, M. (2004). Sparse graphical models for exploring gene expression data. *Journal of Multivariate Analysis*, 90(1):196–212. 149, 150, 151

Dodier, R. and Henze, G. (2004). Statistical analysis of neural networks as applied to building energy prediction. *Journal of solar energy engineering*, 126:592. 30

DOE (2011). Doe releases new version of energyplus modeling software. http://apps1.eere.energy.gov/news/progress_alerts.cfm/pa_id=651. 19

DOE (2012a). Energyplus. http://energyplus.gov. 19

DOE (2012b). Energyplus licensing. http://apps1.eere.energy.gov/buildings/energyplus/energyplus_licensing.cfm. 18

DOE (2012c). Getting started with energyplus: Basic concepts manual - essential information you need about running energyplus. http://apps1.eere.energy.gov/buildings/energyplus/pdfs/gettingstarted.pdf. 19

DOE (2012d). Tips & tricks for using energyplus: Insider secrets to using energyplus. http://apps1.eere.energy.gov/buildings/energyplus/pdfs/tips_and_tricks_using_energyplus.pdf. 19

Dong, B., Cao, C., and Lee, S. (2005). Applying support vector machines to predict building energy consumption in tropical region. *Energy and Buildings*, 37(5):545–553. 16

Edwards, R. E., New, J., and Parker, L. E. (2012). Predicting future hourly residential electrical consumption: A machine learning case study. *Energy and Buildings*, 49:591–603. 25, 116, 176

Feuston, B. and Thurtell, J. (1994). Generalized nonlinear regression with ensemble of neural nets: the great energy predictor shootout. *ASHRAE Transactions.*, (5-1080). 15

Franc, V. and Sonnenburg, S. (2009). Optimized cutting plane algorithm for large-scale risk minimization. *The Journal of Machine Learning Research*, 10:2157–2192. 110

Friedman, N. (1999). Learning bayesian network structure from massive datasets: The sparse candidate algorithm background: Learning structure. *Science*, pages 206–215. 143, 144

Gonzalez, P. and Zamarreno, J. (2005). Prediction of hourly energy consumption in buildings based on a feedback artificial neural network. *Energy and buildings*, 37(6):595–601. 30, 40

Gustafsson, M., Hornquist, M., and Lombardi, A. (2004). Large-scale reverse engineering by the lasso. *Arxiv preprint q-bio/0403012*. 149

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182. 52

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1):389–422. 147

Helton, J., Johnson, J., Sallaberry, C., and Storlie, C. (2006). Survey of sampling-based methods for uncertainty and sensitivity analysis. *Reliability Engineering & System Safety*, 91(10):1175 – 1209. The Fourth International Conference on Sensitivity Analysis of Model Output (SAMO 2004). 13

Hendron, R., Engebrecht, C., and (US), N. R. E. L. (2010). *Building America House Simulation Protocols*. National Renewable Energy Laboratory. http://www.nrel.gov/docs/fy11osti/49246.pdf. 20

Hinneburg, A., Keim, D., et al. (1999). Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In *Proceedings of the 25th VLDB Conference*. 132

Hoegaerts, L., Suykens, J., Vandewalle, J., and De Moor, B. (2004). A comparison of pruning algorithms for sparse least squares support vector machines. In *Neural Information Processing*, pages 1247–1253. Springer. 33

Hong, T., Buhl, F., Haves, P., Selkowitz, S., and Wetter, M. (2008). Comparing computer run time of building simulation programs. http://escholarship.org/uc/item/6504q6d0#page-1. 19

Ibargüengoytia, P. H., Sucar, L. E., and Vadera, S. (2001). Real Time Intelligent Sensor Validation. *IEEE Transactions on Power Systems*, 16(4):770–775. 27

Iijima, M., Takeuchi, R., Takagi, K., and Matsumoto, T. (1994). Piecewise-linear regression on the ashrae time-series data. *ASHRAE Transactions*, 100(2):1088–1095. 15

Jin, R., Chen, W., and Simpson, T. (2001). Comparative studies of metamodelling techniques under multiple modelling criteria. *Structural and Multidisciplinary Optimization*, 23(1):1–13. 13, 105

Joachims, T. (1999a). *Making large-scale SVM learning practical*. MIT press. 23

Joachims, T. (1999b). Making large-Scale SVM Learning Practical. *Advances in Kernel Methods Support Vector Learning*, pages 169–184. 109

Jordan, M., Ghahramani, Z., Jaakkola, T., and Saul, L. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233. 141

Jordan, M. and Jacobs, R. (1992). Hierarchies of Adaptive Experts. *Advances in Neural Information Processing Systems 4*, pages 985–993. 35

Jordan, M. and Jacobs, R. (1994). Hierarchical Mixtures of Experts and the EM Algorithm. *Neural computation*, 6(2):181–214. xiii, 33, 35, 36

Kaminski, T., Heimann, M., Giering, R., et al. (1999). A coarse grid threedimensional global inverse model of the atmospheric transport: 2. inversion of the transport of co2 in the 1980s. *J. Geophys. Res*, 104(18):555–18. 105

Karatasou, S., Santamouris, M., and Geros, V. (2006). Modeling and predicting building's energy use with artificial neural networks: Methods and results. *Energy and buildings*, 38(8):949–958. 16, 24, 30, 40, 42

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. The MIT Press. 141

Kolter, J. and Ferreira Jr, J. (2011). A large-scale study on predicting and contextualizing building energy usage. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*. 17, 28

Kreider, J. and Haberl, J. (1994). Predicting hourly building energy use: the great energy predictor shootout- overview and discussion of results. *ASHRAE Transactions*, 100(2):1104–1118. 14, 24, 40, 42, 116

Larrañaga, P., Poza, M., Yurramendi, Y., Murga, R., and Kuijpers, C. (1996). Structure learning of bayesian networks by genetic algorithms: A performance analysis of control parameters. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(9):912–926. 145

Li, F. (2007). *Structure learning with large sparse undirected graphs and its applications*. PhD thesis, Carnegie Mellon University. 60, 151, 152, 155, 156, 174

Li, K., Su, H., and Chu, J. (2011). Forecasting building energy consumption using neural networks and hybrid neuro-fuzzy system: a comparative study. *Energy and Buildings*. 16, 24, 42, 45

Lima, C., Coelho, A., and Von Zuben, F. (2009). Pattern classification with mixtures of weighted least-squares support vector machine experts. *Neural computing & applications*, 18(7):843–860. 36

MacKay, D. et al. (1994). Bayesian nonlinear modeling for the prediction competition. *Ashrae Transactions*, 100(2):1053–1062. 14, 60

Margaritis, D. and Thrun., S. (1999). Bayesian network induction via local neighborhoods. *Trans. on Pattern Analysis and Machine Intelligence*. 146

Martin, D., Fowlkes, C., and Malik, J. (2004). Learning to detect natural image boundaries using local brightness, color, and texture cues. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(5):530–549. 47

Miller, A. (2002). *Subset selection in regression*, volume 95. CRC Press. 148

Nesterov, Y., Nemirovskii, A., and Ye, Y. (1994). *Interior-point polynomial algorithms in convex programming*, volume 13. SIAM. 109

Ong, Y., Nair, P., and Keane, A. (2003). Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA journal*, 41(4):687–696. 105

Pedrini, A., Westphal, F., and Lamberts, R. (2002). A methodology for building energy modeling and calibration in warm climates. *Building and Environment*, 37(89):903 – 912. 11

Pellet, J. and Elisseeff, A. (2008). Using markov blankets for causal structure learning. *The Journal of Machine Learning Research*, 9:1295–1342. 147, 148

Peter Ellis, B. L. S. (2012). Epx, a modified version of energyplus. http://bigladdersoftware.com/epx/. 18

Qian, Z., Seepersad, C., Joseph, V., Allen, J., and Wu, C. (2006). Building surrogate models based on detailed and approximate simulations. *Transactions-American Society of Mechanical Engineers Journal of Mechanical Design*, 128(4):668. 105

Raftery, P., Keane, M., and ODonnell, J. (2011). Calibrating whole building energy models: An evidence-based methodology. *Energy and Buildings*, 43(9):2356 – 2364. 11

Rissanen, J. (1983). A universal prior for integers and estimation by minimum description length. *The Annals of statistics*, 11(2):416–431. 55

Rumantir, G. (1999). Comparison of second-order polynomial model selection methods: an experimental survey. In *Proceedings of the National Conference on Artificial Intelligence*, pages 980–980. John Wiley & Sons LTD. 55

Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall series in artificial intelligence. Prentice Hall. 27, 143, 145

Sanyal, H., Al-Wadei, Y. H., Bhandari, M. S., Shrestha, S. S., Karpay, B., Garret, A. L., Edwards, R. E., Parker, L. E., and New, J. R. (2012). Poster: Building energy model calibration using energyplus, supercomputing, and machine learning. *Proceedings of the 5th National SimBuild of IBPSA-USA*. 22, 146, 151

Schaller, R. (1997). Moore's law: past, present and future. *IEEE Spectrum*, 34(6):52–59. 26

Schwarz, G. (1978). Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464. 55, 109, 142

Shalev-Shwartz, S. and Srebro, N. (2008). Svm optimization: inverse dependence on training set size. In *Proceedings of the 25th international conference on Machine learning*, pages 928–935. ACM. 23

Smola, A.J. and Schólkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222. 31, 32

Spirtes, P., Glymour, C., and Scheines, R. (1989). *Causality from probability*. Carnegie-Mellon University, Laboratory for Computational Linguistics. 146

Storlie, C. B., Swiler, L. P., Helton, J. C., and Sallaberry, C. J. (2009). Implementation and evaluation of nonparametric regression procedures for

sensitivity analysis of computationally demanding models. *Reliability Engineering & System Safety*, 94(11):1735 – 1763. 13

Suykens, J., Gestel, T. V., Brabanter, J. D., Moor, B. D., and Vandewalle, J. (2002a). *Least squares support vector machines*. World Scientific Pub Co Inc. 33, 47

Suykens, J., J., D. B., L., L., and J., V. (2002b). Weighted least squares support vector machines: robustness and sparse approximation. *Neurocomputing*, 48(1-4):85–105. 33, 36

Teo, C. H., Le, Q., Smola, A., and Vishwanathan, S. V. N. (2007). A scalable modular convex solver for regularized risk minimization. In *KDD. ACM*. 110

Tian, W. and Choudhary, R. (2012). A probabilistic energy model for non-domestic building sectors applied to analysis of school buildings in greater london. *Energy and Buildings*, 54(0):1 – 11. 12, 13, 106

Tipping, M. (2001). Sparse bayesian learning and the relevance vector machine. *The Journal of Machine Learning Research*, 1:211–244. 59

Tresidder, E., Zhang, Y., and Forrester, A. (2011). Optimisation of low-energy building design using surrogate models. *12th Conference of International Building Performance Simulation Association, Syndey AUS*. 105

Tresidder, E., Zhang, Y., and Forrester, A. (2012). Acceleration of building design optimisation through the use of kriging surrogate models. *First Building Simulation and Optimization Conference Loughborough, UK*. 106

Tsamardinos, I., Aliferis, C., and Statnikov, A. (2003). Time and sample efficient discovery of markov blankets and direct causal relations. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 673–678. ACM. 144

Tsamardinos, I., Brown, L., and Aliferis, C. (2006). The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78. 144

Vapnik, V. (1999). An overview of statistical learning theory. *Neural Networks, IEEE Transactions on*, 10(5):988–999. 16

Westphal, F. S. and Lamberts, R. (2005). Building simulation calibration using sensitivity analysis. In *Building Simulation*, volume 9, pages 1331–1338. Citeseer. 12

Willcox, K. and Peraire, J. (2002). Balanced model reduction via the proper orthogonal decomposition. *AIAA journal*, 40(11):2323–2330. 105

Willsky, A., Malioutov, D., et al. (2008). *Approximate inference in Gaussian graphical models*. PhD thesis, Massachusetts Institute of Technology. 150

Yang, J., Rivard, H., and Zmeureanu, R. (2005). On-line building energy prediction using adaptive artificial neural networks. *Energy and buildings*, 37(12):1250–1259. 30

Yoon, J.-H. and Lee, E.-J. (1999). Calibration procedure of energy performance simulation model for a commercial building. In *Proceedings from the Building Simulation Conference*, volume 3, pages 1439–1446. 11

Zeng, X., Drewniak, B. A., and Constantinescu, E. M. (2013). Calibration of the crop model in the community land model. *Geoscientific Model Development Discussions*, 6:379–398. 12

Zhang, H. and Parker, L. (2011). 4-dimensional local spatio-temporal features for human activity recognition. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2044–2049. IEEE. 141

Appendix

Appendix A

Variable Tables

This appendix contains tables that map variable numbers to the real world variable name.

Table 1: MO1 and MO2 input variables 1 - 52.

Variable Number	Name
1	North Axis
2	Terrain
3	Loads Convergence Tolerance Value
4	Temperature Convergence Tolerance Value
5	Solar Distriburtion
6	Maximum Number of Warmup Days
7	Calculation Frequency
8	Maximum Figures in Shadow Overlap Calculations
9	SurfaceConvectionAlgorithm:Inside
10	SurfaceConvectionAlgorithm:Outside
11	Timestep
12	Schedule:Compact IO12
13	Material: Gypsum board 5/8” Smooth: Thickness
14	Material: Gypsum board 5/8” Smooth: Conductivity
15	Material: Gypsum board 5/8” Smooth: Density
16	Material: Gypsum board 5/8” Smooth: Specific Heat
17	Material: Gypsum board 5/8” Smooth: Thermal Absorptance
18	Material: Gypsum board 5/8” Smooth: Solar Absorptance
19	Material: Gypsum board 5/8” Smooth: Visible Absorptance
20	Material:Culture Stone VeryRough: Thickness
21	Material:Culture Stone VeryRough: Conductivity
22	Material:Culture Stone VeryRough: Density
23	Material:Culture Stone VeryRough: Specific Heat
24	Material:Culture Stone VeryRough: Thermal Absorptance
25	Material:Culture Stone VeryRough: Solar Absorptance
26	Material:Culture Stone VeryRough: Visible Absorptance
27	Material:Glass Fiber insulation 2 3/8” Smooth: Thickness
28	Material:Glass Fiber insulation 2 3/8” Smooth: Conductivity
29	Material:Glass Fiber insulation 2 3/8” Smooth: Density
30	Material:Glass Fiber insulation 2 3/8” Smooth: Specific Heat
31	Material:Concrete Foundation Wall Medium Smooth: Thickness
32	Material:Concrete Foundation Wall Medium Smooth: Conductivity
33	Material:Concrete Foundation Wall Medium Smooth: Density
34	Material:Concrete Foundation Wall Medium Smooth: Specific Heat
35	Material:Concrete Foundation Wall Medium Smooth: Thermal Absorptance
36	Material:Concrete Foundation Wall Medium Smooth: Solar Absorptance
37	Material:Concrete Foundation Wall Medium Smooth: Visible Absorptance
38	Material:StandingSeamMetal Roofing Smooth: Thickness
39	Material:StandingSeamMetal Roofing Smooth: Conductivity
40	Material:StandingSeamMetal Roofing Smooth: Density
41	Material:StandingSeamMetal Roofing Smooth: Specific Heat
42	Material:StandingSeamMetal Roofing Smooth: Thermal Absorptance
43	Material:StandingSeamMetal Roofing Smooth: Solar Absorptance
44	Material:StandingSeamMetal Roofing Smooth: Visible Absorptance
45	Material:Plywood 1/2” MediumSmooth: Thickness
46	Material:Plywood 1/2” MediumSmooth: Conductivity
47	Material:Plywood 1/2” MediumSmooth: Density
48	Material:Plywood 1/2” MediumSmooth: Specific Heat
49	Material:Concrete Slab 4” MediumRough: Thickness
50	Material:Concrete Slab 4” MediumRough: Conductivity
51	Material:Concrete Slab 4” MediumRough: Density
52	Material:Concrete Slab 4” MediumRough: Specific Heat

Table 2: MO1 and MO2 input variables 53 - 104.

Variable Number	Name
53	Material:Concrete Slab 4” MediumRough: Thermal Absorptance
54	Material:XPS Ridge Insulation 1” Smooth: Thickness
55	Material:XPS Ridge Insulation 1” Smooth: Conductivity
56	Material:XPS Ridge Insulation 1” Smooth: Density
57	Material:XPS Ridge Insulation 1” Smooth: Specific Heat
58	Material:Gravel 4” VeryRough: Thickness
59	Material:Gravel 4” VeryRough: Conductivity
60	Material:Gravel 4” VeryRough: Density
61	Material:Gravel 4” VeryRough: Specific Heat
62	Material:OSB 7/16” MediumSmooth: Thickness
63	Material:OSB 7/16” MediumSmooth: Conductivity
64	Material:OSB 7/16” MediumSmooth: Density
65	Material:OSB 7/16” MediumSmooth: Specific Heat
66	Material:OSB 7/16” MediumSmooth: Thermal Absorptance
67	Material:EPS 5 + 5/8” MediumSmooth: Thickness
68	Material:EPS 5 + 5/8” MediumSmooth: Conductivity
69	Material:EPS 5 + 5/8” MediumSmooth: Density
70	Material:EPS 5 + 5/8” MediumSmooth: Specific Heat
71	Material:EPS 9 + 3/8” Medium Smooth: Thickness
72	Material:EPS 9 + 3/8” Medium Smooth: Conductivity
73	Material:EPS 9 + 3/8” Medium Smooth: Density
74	Material:EPS 9 + 3/8” Medium Smooth: Specific Heat
75	Material:Hardie Cladding 1/4” Gray MediumRough: Thickness
76	Material:Hardie Cladding 1/4” Gray MediumRough: Conductivity
77	Material:Hardie Cladding 1/4” Gray MediumRough: Density
78	Material:Hardie Cladding 1/4” Gray MediumRough: Specific Heat
79	Material:Hardie Cladding 1/4” Gray MediumRough: Thermal Absorptance
80	Material:Hardie Cladding 1/4” Gray MediumRough: Solar Absorptance
81	Material:Hardie Cladding 1/4” Gray MediumRough: Visible Absorptance
82	Material:Hardie Cladding 1/4” Dark Green MediumRough: Thickness
83	Material:Hardie Cladding 1/4” Dark Green MediumRough: Conductivity
84	Material:Hardie Cladding 1/4” Dark Green MediumRough: Density
85	Material:Hardie Cladding 1/4” Dark Green MediumRough: Specific Heat
86	Material:Hardie Cladding 1/4” Dark Green MediumRough: Thermal Absorptance
87	Material:Hardie Cladding 1/4” Dark Green MediumRough: Solar Absorptance
88	Material:Hardie Cladding 1/4” Dark Green MediumRough: Visible Absorptance
89	Material:NoMass:Building Wrap Smooth: Thermal Resistance
90	Material:AirGap:Dimpled Space Mat: Thermal Resistance
91	WindowMaterial:SimpleGlazing System: Window A: U-Factor
92	WindowMaterial:SimpleGlazing System: Window A: Solar Heat Gain Coefficient
93	WindowMaterial:SimpleGlazing System: Window B: U-Factor
94	WindowMaterial:SimpleGlazing System: Window B: Solar Heat Gain Coefficient
95	WindowMaterial:SimpleGlazing System: Window C: U-Factor
96	WindowMaterial:SimpleGlazing System: Window C: Solar Heat Gain Coefficient
97	WindowMaterial:SimpleGlazing System: Window D: U-Factor
98	WindowMaterial:SimpleGlazing System: Window D: Solar Heat Gain Coefficient
99	WindowMaterial:SimpleGlazing System: Window E: U-Factor
100	WindowMaterial:SimpleGlazing System: Window E: Solar Heat Gain Coefficient
101	WindowMaterial:SimpleGlazing System: Window F: U-Factor
102	WindowMaterial:SimpleGlazing System: Window F: Solar Heat Gain Coefficient
103	WindowMaterial:SimpleGlazing System: Window G: U-Factor
104	WindowMaterial:SimpleGlazing System: Window G: Solar Heat Gain Coefficient

Table 3: MO1 and MO2 input variables 104 - 156.

Variable Number	Name
105	WindowMaterial:SimpleGlazing System: Window H: U-Factor
106	WindowMaterial:SimpleGlazing System: Window H: Solar Heat Gain Coefficient
107	WindowMaterial:SimpleGlazing System: Window I: U-Factor
108	WindowMaterial:SimpleGlazing System: Window I: Solar Heat Gain Coefficient
109	ShadingProperty:Ref lectance:South Overhand Lower Roof East: Diffuse Solar Reflectance
110	ShadingProperty:Ref lectance:South Overhand Lower Roof East: Diffuse Visible Reflectance
111	Lights:Bathroom (1st and 2nd floor): Fraction Radiant
112	Lights:Bathroom (1st and 2nd floor): Fraction Visible
113	Lights:Bedroom (Master and 2nd floor bed rooms): Fraction Radiant
114	Lights:Bedroom (Master and 2nd floor bed rooms): Fraction Visible
115	Lights:Level 1 Lights(Kitchen and Dining): Fraction Radiant
116	Lights:Level 1 Lights(Kitchen and Dining): Fraction Visible
117	ElectricEquipment:Dryer: Fraction Latent
118	ElectricEquipment:Dryer: Fraction Radiant
119	ElectricEquipment:Dryer: Fraction Lost
120	ElectricEquipment:Oven and Microwave: Radiant
121	ElectricEquipment:Dishwasher: Fraction Latent
122	ElectricEquipment:Dishwasher: Fraction Radiant
123	ElectricEquipment:Dishwasher: Fraction Lost
124	ElectricEquipment:Refrigerator: Fraction Radiant
125	ElectricEquipment:Clothes Washer: Fraction Latent
126	ElectricEquipment:Clothes Washer: Fraction Lost
127	ElectricEquipment:Clothes Washer: Fraction Radiant
128	ElectricEquipment:Bathroom Plugs (Heater on 2nd Floor): Fraction Radiant
129	ElectricEquipment:Dining Room Plugs: Fraction Radiant
130	ElectricEquipment:Basement Plugs: Fraction Radiant
131	ElectricEquipment:ERV Energy:Basement: Fraction Radiant
132	ElectricEquipment:ERV Energy:Basement: Fraction Latent
133	ZoneInfiltration:FlowCoefficient:Living: Flow Coefficient
134	ZoneInfiltration:FlowCoefficient:Master Bedroom: Flow Coefficient
135	ZoneInfiltration:FlowCoefficient:Basement: Flow Coefficient
136	ZoneInfiltration:FlowCoefficient:Second Floor: Flow Coefficient
137	ZoneHVAC:IdealLoadAirSystem:Living Room: Maximum Heating Supply Air Temperature
138	ZoneHVAC:IdealLoadAirSystem:Living Room: Minimum Cooling Supply Air Temperature
139	ZoneHVAC:IdealLoadAirSystem:Living Room: Maximum Heating Supply Air Humidity Ratio
140	ZoneHVAC:IdealLoadAirSystem:Living Room: Minimum Cooling Supply Air Humidity Ratio
141	ZoneHVAC:IdealLoadAirSystem:Living Room: Cooling Sensible Heat Ratio
142	ZoneHVAC:IdealLoadAirSystem:Master Bedroom: Maximum Heating Supply Air Temperature
143	ZoneHVAC:IdealLoadAirSystem:Master Bedroom: Minimum Cooling Supply Air Temperature
144	ZoneHVAC:IdealLoadAirSystem:Master Bedroom: Maximum Heating Supply Air Humidity Ratio
145	ZoneHVAC:IdealLoadAirSystem:Master Bedroom: Minimum Cooling Supply Air Humidity Ratio
146	ZoneHVAC:IdealLoadAirSystem:Master Bedroom: Cooling Sensible Heat Ratio
147	ZoneHVAC:IdealLoadAirSystem:Basement: Maximum Heating Supply Air Temperature
148	ZoneHVAC:IdealLoadAirSystem:Basement: Minimum Cooling Supply Air Temperature
149	ZoneHVAC:IdealLoadAirSystem:Basement: Maximum Heating Supply Air Humidity Ratio
150	ZoneHVAC:IdealLoadAirSystem:Basement: Minimum Cooling Supply Air Humidity Ratio
151	ZoneHVAC:IdealLoadAirSystem:Basement: Cooling Sensible Heat Ratio
152	ZoneHVAC:IdealLoadAirSystem:Second Floor: Maximum Heating Supply Air Temperature
153	ZoneHVAC:IdealLoadAirSystem:Second Floor: Minimum Cooling Supply Air Temperature
154	ZoneHVAC:IdealLoadAirSystem:Second Floor: Maximum Heating Supply Air Humidity Ratio
155	ZoneHVAC:IdealLoadAirSystem:Second Floor: Minimum Cooling Supply Air Humidity Ratio
156	ZoneHVAC:IdealLoadAirSystem:Second Floor: Cooling Sensible Heat Ratio

Table 4: MO1 and MO2 output variables 1 - 40

Variable Number	Name
1	BATHROOM LIGHTS (1ST AND 2ND FLOOR)ENERGY:Lights Electric Power
2	BATHROOM LIGHTS (1ST AND 2ND FLOOR)ENERGY:Lights Total Heat Gain Rate
3	BEDROOM LIGHTS AND PLUGS (MASTER AND 2ND FLOOR BED ROOMS) ENERGY:Lights Electric Power
4	BEDROOM LIGHTS AND PLUGS (MASTER AND 2ND FLOOR BED ROOMS) ENERGY:Lights Total Heat Gain Rate
5	LEVEL 1 LIGHTS (KITCHEN AND DINING) ENERGY:Lights Electric Power
6	LEVEL 1 LIGHTS (KITCHEN AND DINING) ENERGY:Lights Total Heat Gain Rate
7	WAHP AUXILLIARY HEAT ENERGY:Electric Equipment Total Heat Gain Rate
8	IHP INDOOR FAN ENERGY:Electric Equipment Total Heat Gain Rate
9	DRYER ENERGY:Electric Equipment Electric Power
10	DRYER ENERGY:Electric Equipment Total Heat Gain Rate
11	OVEN/MICROWAVE ENERGY:Electric Equipment Electric Power
12	OVEN/MICROWAVE ENERGY:Electric Equipment Total Heat Gain Rate
13	DISHWASHER ENERGY:Electric Equipment Electric Power
14	DISHWASHER ENERGY:Electric Equipment Total Heat Gain Rate
15	REFRIGERATOR ENERGY:Electric Equipment Electric Power
16	REFRIGERATOR ENERGY:Electric Equipment Total Heat Gain Rate
17	KITCHEN PLUGS ENERGY:Electric Equipment Electric Power
18	KITCHEN PLUGS ENERGY:Electric Equipment Total Heat Gain Rate
19	CLOTHER WASHER ENERGY:Electric Equipment Electric Power
20	CLOTHER WASHER ENERGY:Electric Equipment Total Heat Gain Rate
21	BATHROOM PLUGS (HEATER ON 2ND FLOOR) ENERGY:Electric Equipment Electric Power
22	BATHROOM PLUGS (HEATER ON 2ND FLOOR) ENERGY:Electric Equipment Total Heat Gain Rate
23	DINNING ROOM PLUGS (1ST FLOOR HEATER) ENERGY:Electric Equipment Electric Power
24	DINNING ROOM PLUGS (1ST FLOOR HEATER) ENERGY:Electric Equipment Total Heat Gain Rate
25	BASEMENT PLUGS ENERGY:Electric Equipment Electric Power
26	BASEMENT PLUGS ENERGY:Electric Equipment Total Heat Gain Rate
27	ERV ENERGY:Electric Equipment Electric Power
28	ERV ENERGY:Electric Equipment Total Heat Gain Rate
29	WAHP:Electric Equipment Total Heat Gain Rate
30	WWHP LOAD:Hot Water Equipment Total Heat Gain Rate
31	LIVING WINDOW NORTH A2:Window Heat Gain
32	LIVING WINDOW NORTH A2:Window Heat Loss
33	NORTH ROOF HFT:Surface Ext Solar Incident
34	MB WINDOW SOUTH A1:Window Heat Gain
35	MB WINDOW SOUTH A1:Window Heat Loss
36	SOUTH ROOF HFT:Surface Ext Solar Beam Incident
37	STAIRWELL EAST WINDOW H1:Surface Inside Temperature
38	STAIRWELL EAST WINDOW H1:Surface Outside Temperature
39	FOYER NORTH WALL HFT:Surface Inside Temperature
40	FOYER NORTH WALL HFT:Surface Outside Temperature

Table 5: MO1 and MO2 output variables 41 - 90

Variable Number	Name
41	FOYER NORTH WALL HFT:Opaque Surface Inside Face Conduction
42	LIVING WINDOW NORTH A2:Surface Inside Temperature
43	LIVING WINDOW NORTH A2:Surface Outside Temperature
44	LIVING WINDOW WEST A2:Surface Inside Temperature
45	LIVING WINDOW WEST A2:Surface Outside Temperature
46	LIVING ROOM WEST WALL HFT:Opaque Surface Inside Face Conduction
47	NORTH ROOF HFT:Surface Inside Temperature
48	NORTH ROOF HFT:Surface Outside Temperature
49	NORTH ROOF HFT:Opaque Surface Inside Face Conduction
50	MB SOUTH WALL HFT:Surface Inside Temperature
51	MB SOUTH WALL HFT:Surface Outside Temperature
52	MB SOUTH WALL HFT:Opaque Surface Inside Face Conduction
53	MB WINDOW SOUTH A1:Surface Inside Temperature
54	MB WINDOW SOUTH A1:Surface Outside Temperature
55	MB EAST WALL HFT:Opaque Surface Inside Face Conduction
56	BM SOUTH WALL EAST:Surface Inside Temperature
57	BM SOUTH WALL EAST:Surface Outside Temperature
58	SOUTH ROOF HFT:Surface Inside Temperature
59	SOUTH ROOF HFT:Surface Outside Temperature
60	SOUTH ROOF HFT:Opaque Surface Inside Face Conduction
61	LIVING:Zone Mean Radiant Temperature
62	LIVING:Zone Mean Air Temperature
63	MASTER BEDROOM:Zone Mean Air Temperature
64	BASEMENT:Zone Mean Air Temperature
65	SECOND FLOOR:Zone Mean Air Temperature
66	LIVING:Zone Infiltration Air Change Rate
67	MASTER BEDROOM:Zone Infiltration Air Change Rate
68	BASEMENT:Zone Infiltration Air Change Rate
69	SECOND FLOOR:Zone Infiltration Air Change Rate
70	LIVING:Zone Air Relative Humidity
71	MASTER BEDROOM:Zone Air Relative Humidity
72	BASEMENT:Zone Air Relative Humidity
73	SECOND FLOOR:Zone Air Relative Humidity
74	LIVING ROOM ZONE IDEAL LOADS AIR:Ideal Loads Sensible Heating Energy
75	LIVING ROOM ZONE IDEAL LOADS AIR:Ideal Loads Latent Heating Energy
76	LIVING ROOM ZONE IDEAL LOADS AIR:Ideal Loads Sensible Cooling Energy
77	LIVING ROOM ZONE IDEAL LOADS AIR:Ideal Loads Latent Cooling Energy
78	MB ZONE IDEAL LOADS AIR:Ideal Loads Sensible Heating Energy
79	MB ZONE IDEAL LOADS AIR:Ideal Loads Latent Heating Energy
80	MB ZONE IDEAL LOADS AIR:Ideal Loads Sensible Cooling Energy
81	MB ZONE IDEAL LOADS AIR:Ideal Loads Latent Cooling Energy
82	BASEMENT ZONE IDEAL LOADS AIR:Ideal Loads Sensible Heating Energy
83	BASEMENT ZONE IDEAL LOADS AIR:Ideal Loads Latent Heating Energy
84	BASEMENT ZONE IDEAL LOADS AIR:Ideal Loads Sensible Cooling Energy
85	BASEMENT ZONE IDEAL LOADS AIR:Ideal Loads Latent Cooling Energy
86	SF ZONE IDEAL LOADS AIR:Ideal Loads Sensible Heating Energy
87	SF ZONE IDEAL LOADS AIR:Ideal Loads Latent Heating Energy
88	SF ZONE IDEAL LOADS AIR:Ideal Loads Sensible Cooling Energy
89	SF ZONE IDEAL LOADS AIR:Ideal Loads Latent Cooling Energy
90	Whole Building:Total Building Electric Demand

Table 6: FG input variables 1 - 50

Variable Number	Name
1	ZEBRAllianceHouseNo 1 SIP House:North Axis
2	ZEBRAllianceHouseNo 1 SIP House:Terrain
3	ZEBRAllianceHouseNo 1 SIP House:LoadsConvergenceToleranceValue
4	ZEBRAllianceHouseNo 1 SIP House:TempConvergenceToleranceValue
5	ZEBRAllianceHouseNo 1 SIP House:Solar Distribution
6	ZEBRAllianceHouseNo 1 SIP House:Maximum Number of Warmup Days
7	ShadowCalculation:Calculation Frequency
8	ShadowCalculation:Max Figures in Shadow Overlap Calculations
9	SurfaceConvectionAlgorithm:Inside:Algorithm
10	SurfaceConvectionAlgorithm:Outside:Algorithm
11	Timestep:Number of Timesteps per Hour
12	Overhang:Field 4
13	Gypsum Board 5/8:Thickness
14	Gypsum Board 5/8:Conductivity
15	Gypsum Board 5/8:Density
16	Gypsum Board 5/8:Specific Heat
17	Gypsum Board 5/8:Thermal Absorptance
18	Gypsum Board 5/8:Solar Absorptance
19	Gypsum Board 5/8:Visible Absorptance
20	Cultured Stone:Thickness
21	Cultured Stone:Conductivity
22	Cultured Stone:Density
23	Cultured Stone:Specific Heat
24	Cultured Stone:Thermal Absorptance
25	Cultured Stone:Solar Absorptance
26	Cultured Stone:Visible Absorptance
27	Glass Fiber Insulation 2 3/8:Thickness
28	Glass Fiber Insulation 2 3/8:Conductivity
29	Glass Fiber Insulation 2 3/8:Density
30	Glass Fiber Insulation 2 3/8:Specific Heat
31	Concrete Foundation Wall:Thickness
32	Concrete Foundation Wall:Conductivity
33	Concrete Foundation Wall:Density
34	Concrete Foundation Wall:Specific Heat
35	Concrete Foundation Wall:Thermal Absorptance
36	Concrete Foundation Wall:Solar Absorptance
37	Concrete Foundation Wall:Visible Absorptance
38	Standing Seam Metal Roofing:Thickness
39	Standing Seam Metal Roofing:Conductivity
40	Standing Seam Metal Roofing:Density
41	Standing Seam Metal Roofing:Specific Heat
42	Standing Seam Metal Roofing:Thermal Absorptance
43	Standing Seam Metal Roofing:Solar Absorptance
44	Standing Seam Metal Roofing:Visible Absorptance
45	Plywood 1/2:Thickness
46	Plywood 1/2:Conductivity
47	Plywood 1/2:Density
48	Plywood 1/2:Specific Heat
49	2x6 Wood Studs:Thickness
50	2x6 Wood Studs:Conductivity

Table 7: FG input variables 51 - 100

Variable Number	Name
51	2x6 Wood Studs:Density
52	2x6 Wood Studs:Specific Heat
53	2x6 Wood Studs:Thermal Absorptance
54	3/4 T&G Plywood Decking:Thickness
55	3/4 T&G Plywood Decking:Conductivity
56	3/4 T&G Plywood Decking:Density
57	3/4 T&G Plywood Decking:Specific Heat
58	3/4 T&G Plywood Decking:Thermal Absorptance
59	3/4 T&G Plywood Decking:Solar Absorptance
60	3/4 T&G Plywood Decking:Visible Absorptance
61	Concrete Slab 4:Thickness
62	Concrete Slab 4:Conductivity
63	Concrete Slab 4:Density
64	Concrete Slab 4:Specific Heat
65	Concrete Slab 4:Thermal Absorptance
66	XPS Rigid Insulation 1:Thickness
67	XPS Rigid Insulation 1:Conductivity
68	XPS Rigid Insulation 1:Density
69	XPS Rigid Insulation 1:Specific Heat
70	Gravel 4:Thickness
71	Gravel 4:Conductivity
72	Gravel 4:Density
73	Gravel 4:Specific Heat
74	OSB 7/16:Thickness
75	OSB 7/16:Conductivity
76	OSB 7/16:Density
77	OSB 7/16:Specific Heat
78	OSB 7/16:Thermal Absorptance
79	EPS 5 + 5/8:Thickness
80	EPS 5 + 5/8:Conductivity
81	EPS 5 + 5/8:Density
82	EPS 5 + 5/8:Specific Heat
83	EPS 9 + 3/8:Thickness
84	EPS 9 + 3/8:Conductivity
85	EPS 9 + 3/8:Density
86	EPS 9 + 3/8:Specific Heat
87	Hardie Cladding 1/4 Gray:Thickness
88	Hardie Cladding 1/4 Gray:Conductivity
89	Hardie Cladding 1/4 Gray:Density
90	Hardie Cladding 1/4 Gray:Specific Heat
91	Hardie Cladding 1/4 Gray:Thermal Absorptance
92	Hardie Cladding 1/4 Gray:Solar Absorptance
93	Hardie Cladding 1/4 Gray:Visible Absorptance
94	Hardie Cladding 1/4 Dark Green:Thickness
95	Hardie Cladding 1/4 Dark Green:Conductivity
96	Hardie Cladding 1/4 Dark Green:Density
97	Hardie Cladding 1/4 Dark Green:Specific Heat
98	Hardie Cladding 1/4 Dark Green:Thermal Absorptance
99	Hardie Cladding 1/4 Dark Green:Solar Absorptance
100	Hardie Cladding 1/4 Dark Green:Visible Absorptance

Table 8: FG input variables 101 - 150

Variable Number	Name
101	Hardie Cladding 1/4 Cream:Thickness
102	Hardie Cladding 1/4 Cream:Conductivity
103	Hardie Cladding 1/4 Cream:Density
104	Hardie Cladding 1/4 Cream:Specific Heat
105	Hardie Cladding 1/4 Cream:Thermal Absorptance
106	Hardie Cladding 1/4 Cream:Solar Absorptance
107	Hardie Cladding 1/4 Cream:Visible Absorptance
108	Building Wrap:Thermal Resistance
109	Dimpled Spacer Mat:Thermal Resistance
110	Window A:U-Factor
111	Window A:Solar Heat Gain Coefficient
112	Window B:U-Factor
113	Window B:Solar Heat Gain Coefficient
114	Window C:U-Factor
115	Window C:Solar Heat Gain Coefficient
116	Window D:U-Factor
117	Window D:Solar Heat Gain Coefficient
118	Window E:U-Factor
119	Window E:Solar Heat Gain Coefficient
120	Window F:U-Factor
121	Window F:Solar Heat Gain Coefficient
122	Window G:U-Factor
123	Window G:Solar Heat Gain Coefficient
124	Window H:U-Factor
125	Window H:Solar Heat Gain Coefficient
126	Window I:U-Factor
127	Window I:Solar Heat Gain Coefficient
128	SouthOverhangLowerRoofEast:DiffSolarReflUnglazedPartShadingSurf
129	SouthOverhangLowerRoofEast:DiffVisibReflUnglazedPartShadingSurf
130	Bathroom Lights (1st and 2nd floor),Energy:Fraction Radiant
131	Bathroom Lights (1st and 2nd floor),Energy:Fraction Visible
132	Bedroom LightsPlugs(master n 2 flr bedrm)Energy:FractionRadiant
133	Bedroom LightsPlugs(master n 2 flr bedrm)Energy:FractionVisible
134	Level 1 Lights (kitchen and dining), Energy:Fraction Radiant
135	Level 1 Lights (kitchen and dining), Energy:Fraction Visible
136	Dryer Energy:Fraction Latent
137	Dryer Energy:Fraction Radiant
138	Dryer Energy:Fraction Lost
139	Oven/Microwave Energy:Fraction Radiant
140	Dishwasher Energy:Fraction Latent
141	Dishwasher Energy:Fraction Radiant
142	Dishwasher Energy:Fraction Lost
143	Refrigerator Energy:Fraction Radiant
144	Clother Washer Energy:Fraction Latent
145	Clother Washer Energy:Fraction Radiant
146	Clother Washer Energy:Fraction Lost
147	Bathroom Plugs (Heater on 2nd floor), Energy:Fraction Radiant
148	Diding Room Plugs (1st floor heater), Energy:Fraction Radiant
149	Basement Plugs Energy:Fraction Radiant
150	ERV Energy:Fraction Radiant

Table 9: FG input variables 151 - 180

Variable Number	Name
151	ERV Energy:Fraction Lost
152	WWHP:Fraction Latent
153	WWHP:Fraction Radiant
154	WWHP:Fraction Lost
155	WAHP:Fraction Radiant
156	WAHP:Fraction Lost
157	Infiltration Living Zone:Flow Coefficient
158	Infiltration Master Bedroom:Flow Coefficient
159	Infiltration Basement:Flow Coefficient
160	Infiltration Second Floor:Flow Coefficient
161	LivingRoomZoneIdealLoadsAir:MaxHeatingSupplyAirTemp
162	LivingRoomZoneIdealLoadsAir:MinCoolingSupplyAirTemp
163	LivingRoomZoneIdealLoadsAir:MaxHeatingSupplyAirHumidityRatio
164	LivingRoomZoneIdealLoadsAir:MinCoolingSupplyAirHumidityRatio
165	LivingRoomZoneIdealLoadsAir:CoolingSensibleHeatRatio
166	MBZoneIdealLoadsAir:MaxHeatingSupplyAirTemp
167	MBZoneIdealLoadsAir:MinCoolingSupplyAirTemp
168	MBZoneIdealLoadsAir:MaxHeatingSupplyAirHumidityRatio
169	MBZoneIdealLoadsAir:MinCoolingSupplyAirHumidityRatio
170	MBZoneIdealLoadsAir:CoolingSensibleHeatRatio
171	BasementZoneIdealLoadsAir:MaxHeatingSupplyAirTemp
172	BasementZoneIdealLoadsAir:MinCoolingSupplyAirTemp
173	BasementZoneIdealLoadsAir:MaxHeatingSupplyAirHumidityRatio
174	BasementZoneIdealLoadsAir:MinCoolingSupplyAirHumidityRatio
175	BasementZoneIdealLoadsAir:CoolingSensibleHeatRatio
176	SFZoneIdealLoadsAir:MaxHeatingSupplyAirTemp
177	SFZoneIdealLoadsAir:MinCoolingSupplyAirTemp
178	SFZoneIdealLoadsAir:MaxHeatingSupplyAirHumidityRatio
179	SFZoneIdealLoadsAir:MinCoolingSupplyAirHumidityRatio
180	SFZoneIdealLoadsAir:CoolingSensibleHeatRatio

Table 10: FG output variables 1 - 40

Variable Number	Name
1	Environment:Outdoor Dry Bulb
2	Environment:Outdoor Relative Humidity
3	Environment:Outdoor Barometric Pressure
4	Environment:Wind Speed
5	Environment:Liquid Precipitation
6	BATHRM LIGHTS(1 N 2ND FLR)ENERGY:Lights Total Heat Gain Rate
7	BEDRM LIGHTS PLUGS(MASTR 2 FLR)ENRGY:Lights Total Heat Gain Rate
8	WAHP AUXILARY HEAT ENRGY:Electric Equipment Total Heat Gain Rate
9	IHP INDOOR FAN ENERGY:Electric Equipment Total Heat Gain Rate
10	DRYER ENERGY:Electric Equipment Total Heat Gain Rate
11	OVEN/MICROWAVE ENERGY:Electric Equipment Total Heat Gain Rate
12	DISHWASHER ENERGY:Electric Equipment Total Heat Gain Rate
13	REFRIGERATOR ENERGY:Electric Equipment Total Heat Gain Rate
14	KITCHEN PLUGS ENERGY:Electric Equipment Total Heat Gain Rate
15	CLOTHER WASHER ENERGY:Electric Equipment Total Heat Gain Rate
16	BATH PLGS(HEATR 2 FLR)ENRGY:ElectricEquipment TotalHeatGain Rate
17	DININGRM PLG(1FLR HEATR)ENRGY:ElectricEquipmentTotalHeatGainRate
18	BASEMENT PLUGS ENERGY:Electric Equipment Total Heat Gain Rate
19	ERV ENERGY:Electric Equipment Total Heat Gain Rate
20	WWHP:Hot Water Equipment Total Heat Gain Rate
21	WAHP:Other Equipment Total Heat Gain Rate
22	LIVING WINDOW NORTH A2:Window Heat Gain
23	LIVING WINDOW NORTH A2:Window Heat Loss
24	NORTH ROOF HFT:Surface Ext Solar Incident
25	MB WINDOW SOUTH A1:Window Heat Gain
26	MB WINDOW SOUTH A1:Window Heat Loss
27	SOUTH ROOF HFT:Surface Ext Solar Beam Incident
28	STAIRWELL EAST WINDOW H1:Surface Inside Temperature
29	STAIRWELL EAST WINDOW H1:Surface Outside Temperature
30	FOYER NORTH WALL HFT:Surface Inside Temperature
31	FOYER NORTH WALL HFT:Surface Outside Temperature
32	FOYER NORTH WALL HFT:Opaque Surface Inside Face Conduction
33	LIVING WINDOW NORTH A2:Surface Inside Temperature
34	LIVING WINDOW NORTH A2:Surface Outside Temperature
35	LIVING WINDOW WEST A2:Surface Inside Temperature
36	LIVING WINDOW WEST A2:Surface Outside Temperature
37	LIVING ROOM WEST WALL HFT:Opaque Surface Inside Face Conduction
38	NORTH ROOF HFT:Surface Inside Temperature
39	NORTH ROOF HFT:Surface Outside Temperature
40	NORTH ROOF HFT:Opaque Surface Inside Face Conduction

Table 11: FG input variables 41 - 80

Variable Number	Name
41	MB SOUTH WALL HFT:Surface Inside Temperature
42	MB SOUTH WALL HFT:Surface Outside Temperature
43	MB SOUTH WALL HFT:Opaque Surface Inside Face Conduction
44	MB WINDOW SOUTH A1:Surface Inside Temperature
45	MB WINDOW SOUTH A1:Surface Outside Temperature
46	MB EAST WALL HFT:Opaque Surface Inside Face Conduction
47	BM SOUTH WALL EAST:Surface Inside Temperature
48	BM SOUTH WALL EAST:Surface Outside Temperature
49	SOUTH ROOF HFT:Surface Inside Temperature
50	SOUTH ROOF HFT:Surface Outside Temperature
51	SOUTH ROOF HFT:Opaque Surface Inside Face Conduction
52	LIVING:Zone Mean Radiant Temperature
53	LIVING:Zone Mean Air Temperature
54	MASTER BEDROOM:Zone Mean Air Temperature
55	BASEMENT:Zone Mean Air Temperature
56	SECOND FLOOR:Zone Mean Air Temperature
57	LIVING:Zone Infiltration Air Change Rate
58	MASTER BEDROOM:Zone Infiltration Air Change Rate
59	BASEMENT:Zone Infiltration Air Change Rate
60	SECOND FLOOR:Zone Infiltration Air Change Rate
61	LIVING:Zone Air Relative Humidity
62	MASTER BEDROOM:Zone Air Relative Humidity
63	BASEMENT:Zone Air Relative Humidity
64	SECOND FLOOR:Zone Air Relative Humidity
65	LIVING RM ZONE IDEAL LOADS AIR:IdealLoadsSensibleHeating Energy
66	LIVING RM ZONE IDEAL LOADS AIR:IdealLoadsLatentHeating Energy
67	LIVING RM ZONE IDEAL LOADS AIR:IdealLoadsSensibleCooling Energy
68	LIVING RM ZONE IDEAL LOADS AIR:IdealLoadsLatentCooling Energy
69	MB ZONE IDEAL LOADS AIR:IdealLoadsSensibleHeating Energy
70	MB ZONE IDEAL LOADS AIR:IdealLoadsLatentHeating Energy
71	MB ZONE IDEAL LOADS AIR:IdealLoadsSensibleCooling Energy
72	MB ZONE IDEAL LOADS AIR:IdealLoadsLatentCooling Energy
73	BASEMENT ZONE IDEAL LOADS AIR:IdealLoadsSensibleHeating Energy
74	BASEMENT ZONE IDEAL LOADS AIR:IdealLoadsLatent Heating Energy
75	BASEMENT ZONE IDEAL LOADS AIR:IdealLoadsSensibleCooling Energy
76	BASEMENT ZONE IDEAL LOADS AIR:IdealLoadsLatentCooling Energy
77	SF ZONE IDEAL LOADS AIR:IdealLoadsSensibleHeating Energy
78	SF ZONE IDEAL LOADS AIR:IdealLoadsLatentHeating Energy
79	SF ZONE IDEAL LOADS AIR:IdealLoadsSensibleCooling Energy
80	SF ZONE IDEAL LOADS AIR:IdealLoadsLatentCooling Energy

Vita

Richard E. Edwards is a Graduate Research Assistant under Dr. Lynne E. Parker in the Distributed Intelligence Laboratory. All work for his PhD dissertation was funded through Dr. Joshua New from Oak Ridge National Lab's Whole Building & Community Integration Group. After completing his PhD in Spring 2013, Richard will relocate to Seattle Washington to work at Amazon.

Richard received his Bachelors of Science in Computer Science from the University of Texas at Austin in Spring 2007. Afterwards, he moved to Knoxville Tennessee to pursue a Masters in Computer Science, which he completed in Spring 2010 with heavy focus on machine learning. After completing his masters, he stayed at the University of Tennessee to work on his doctorate degree under Dr. Parker.