Estimating Building Simulation Parameters via Bayesian Structure Learning

Richard E. Edwards
Electrical Engineering and Computer Science
University of Tennessee
Knoxville TN, United States
redwar15@eecs.utk.edu

Joshua R. New
Whole Building & Community Integration Group
Oak Ridge National Lab
Oak Ridge TN, United States
newjr@ornl.gov

Lynne E. Parker
Electrical Engineering and Computer Science
University of Tennessee
Knoxville TN, United States
parker@eecs.utk.edu

ABSTRACT

Many key building design policies are made using sophisticated computer simulations such as EnergyPlus (E+), the DOE flagship whole-building energy simulation engine. E+ and other 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. Currently, these problems are addressed by having an engineer manually calibrate simulation parameters to real world data or using algorithmic optimization methods to adjust the building parameters. However, some simulations engines, like E+, are computationally expensive, which makes repeatedly evaluating the simulation engine costly. This work explores addressing this issue by automatically discovering the simulation's internal input and output dependencies from ∼20 Gigabytes of E+ simulation data, future extensions will use ∼200 Terabytes of E+ simulation data. The model is validated by inferring building parameters for E+ simulations with ground truth building parameters. Our results indicate that the model accurately represents parameter means with some deviation from the means, but does not support inferring parameter values that exist on the distribution's tail.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Applications and Expert Systems

Keywords

Structure Learning, Probabilistic Inference, Big Data

1. INTRODUCTION

EnergyPlus 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. E+ consists of ∼600k lines of Fortran code, and 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 paper. The computational costs of these capabilities has resulted in annual building simulations that, depending on the complexity of the building information, often require 5+ minutes (10x-100x slower than DOE-2 [11]) of wall-clock time to complete; reducing E+'s runtime is a top priority for its development teams, E+ 7.0 is 25%-40% faster than E+ 6.0.

Even with a 40% reduction in runtime, manually tuning E+ building models is still a very slow and tedious process. For example, an engineer manually calibrating a simulation is not likely to wait the 3-7 minutes required to run an E+ simulation before proceeding to the next calibration step; likewise, optimizing the simulation parameters automatically using ∼1000 simulations, at 3 minutes per simulation, requires over 2 days on a standard desktop computer. However at its core, calibrating an E+ simulation model requires understanding how all the input variables interact and the effects these variables have on the simulation output. This implies it is possible to construct models that can produce quick candidate simulation estimates without optimizing against the simulation engine at all.

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:

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

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 identi-

BigMine'13, August 11, 2013 Chicago, Illinois, USA

cally distributed. Eq. 1 may be rewritten as:

$$\Pi_{i=1}^{N} P(X, Y_i) \tag{2}$$

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

$$\Pi_{i=1}^{N} P(Y_i|X)P(X) \tag{3}$$

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 statistical surrogate models. Therefore, we are only focusing on estimating building parameters and only present this capability for completeness.

2. PROBABILISTIC GRAPHICAL MODEL

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. 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 by splitting it into simpler components. These types of models have been effectively applied to many fields, such as topic classification [2], document classification [1], and disease diagnosis [13].

The graphs used to represent factorized joint probability distributions are either direct acyclic graphs (DAG) or undirected graphs. 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 separated by variables Z. X and Y are said to be separated by Z if and only if all paths between X and Y pass through the variables Z [14]. 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. Additionally, both graphs have weaknesses with representing certain conditional independence properties. However within this work, we show preference to Markov Networks over Bayesian Networks because we wish to avoid falsely representing variables as conditionally independent.

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 address this problem. The first method is Score and Search (Section 3.1), and is generally used to learn Bayesian Network structures. The second approach is Constraint Based (Section 3.2) methods, which are used to learn Bayesian and Markov Networks. The final method uses strong distribution assumptions combined with regression based feature selection methods to determine dependencies among the random variables (Section 3.3).

3. STRUCTURE LEARNING

3.1 Score and Search

Score and Search (S&S) 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) [22]:

$$\text{BIC(Data; G)} = \mathcal{L}(Data; G, \theta) + \frac{\log M}{2} * Dim[G] \tag{4}$$

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 S&S methods are generally used to 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 S&S within the literature is Greedy Hill Climbing, which explores the candidate graph space by deleting edges, adding edges, and changing edge directions. The best valid graph modification is selected according to a criteria function, generally BIC, and a new candidate graph is generated. A modification is only valid if the candidate graph is a valid Bayesian Network. This approach guarantees 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 algorithm by constraining the search space, which allows for better solutions. The first algorithm is the Sparse Candidate method [8]. 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:

$$\mathcal{I}(X, Y) = \sum_{x,y} P(x, y) log\left(\frac{P(x, y)}{P(x)P(y)}\right) \tag{5}$$

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 Sparse Candidate 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 allows the greedy algorithm to select 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 scales well to large Bayesian Networks with hundreds of random variables, but the runtime and solution quality tradeoff are strongly determined by k's selection [26].

The second algorithm, Max-Min Hill-Climb (MMHC [26]), extends the basic greedy hill climbing algorithm by using a Constraint Based (C&B) algorithm (Section 3.2) called Max-Min Parents and Children (MMPC [25]) 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. This algorithm improves on the Sparse Candidate algorithm by providing a tighter upper bound on runtime. In addition, this algorithm 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, which leads to an exponential worst case runtime.

There are many more S&S methods that have been applied to structure learning, such as genetic algorithms [15], best first search, and equivalence class searches [4]. While some S&S methods scale well to large datasets, these methods involve optimizing a BIC, likelihood, or posterior probability criteria function, which does not scale 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 Bayesian Networks may under represent the data's underlying dependencies.

3.2 Constraint Based

Constraint based (CB) 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 CB approaches are only dependent upon statistical testing, they are better suited for learning Markov Networks than the S&S algorithms. This benefit is derived from the fact that these methods do not need to compute a global criteria function.

However, this can also be viewed as a drawback. The problem is best understood by analyzing the simplest CB algorithm, SGS [23], 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 CB algorithms perform well and 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 [18]. 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 does not scale well on E+ data for two reasons: the size of the E+ dataset (millions of data vectors) and the total number of random variables being modeled ($\sim$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.

J. Pellet et. al, [21] 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 [21] proves that if an algorithm has the exact Markov blanket for each random variable, then the algorithm will find the true causal model. A causal model is a Bayesian Network in which edges imply causality, which represents a stronger probabilistic relationship.

While it is computationally intractable to extract the exact Markov blanket for each random variable in large problems, the algorithm does not use dependency test, like GS, to determine the Markov blanket. Rather, the algorithm uses feature selection techniques to determine the Markov blanket for each random variable by using backward-selection based linear regression and stepwise selection linear regression. J. Pellet et. al, [21] also explored a backward-selection method combined with SVM regression, called Recursive Feature Elimination (RFE) [10], but determined that the method is too computationally expensive even though it will detect nonlinear dependencies within the Markov blanket. While the proposed algorithm in [21] is more computationally appealing than the existing CB and S&S methods, it must assume that all regression error terms follow a Gaus-

sian distribution, which makes it slightly less general than the other methods. In addition, other works illustrates that there are better feature selection methods than stepwise selection and backwards selection [19].

While the feature selection methods can be improved, the overall approach is much more scalable than the S&S and CB methods. This clear computational advantage lead us to explore additional Regression based (RB) structure learning methods. In addition, the RB methods presented in this paper have a polynomial worst case runtime, while all other methods have an exponential worst case runtime.

3.3 Regression Based

RB structure learning methods determine all conditional dependencies 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.

The RB approach is extremely scalable. For example, M. Gustafsson et. al [9] used RB methods 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 research, 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 variable and the predictors with a non-zero regression coefficient. While the method directly extracts an undirected conditional dependency graph, it may not be possible to extract an overall joint distribution from the resulting graph.

A. Dobra et. al [5] focuses on recovering the factorized graph's joint distribution. [5] 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:

$$\frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} exp(-\frac{1}{2}(x-\mu)^T\Omega(x-\mu)) \tag{6}$$

where Σ is the covariance matrix, μ is the mean, and Ω is the inverse covariance matrix or precision matrix. The approach learns a Bayesian Network that is converted to a GGM by using the regression coefficients and the variance for each regression to compute the precision matrix Ω. The precision matrix is recovered by the following computation:

$$\Omega = (1-\Gamma)^T\Psi^{-1}(1-\Gamma) \tag{7}$$

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 learning Ω from the data directly, either by estimating Σ and computing Σ^{-1}. The inverse operation requires an $\mathcal{O}(V^3)$ runtime, where V is the total number of random variables, which is not scalable to larger problems. More importantly, directly learning Ω via [7] only guarantees a global maximum solution if there are sufficient observations for the problems dimensionality. However, the method presented by Fan Li [16] guarantees a global maximum solution in all instances.

This model allows exact inference by using the GGM's distribution's information form (Eq. 8).

$$p(x) \propto exp(x^TJx + h^Tx) \tag{8}$$

J is the Ω and h is the potential vector. These methods assume h represents a vector of zeros. Based on the research presented in [27], it is possible to relax the assumption that h is zero. However, computing h requires computing Σ by inverting Ω, which is expensive even with the efficient method presented by A. Dobra et. al [5].

Overall, A. Dobra et. al [5]'s method is fairly robust and computationally feasible for a large number of random variables. However, the resulting model is heavily dependent upon the variable ordering. Each unique variable ordering may produce a different joint probability distribution because each variable is only dependent upon variables that proceed it. This assumption ensures that the resulting graph structure is a valid Bayesian Network, but requires the method to search the set of all possible orders to determine the best structure. A. Dobra et. al [5] addresses this issue by scoring the individual variables and greedily ordering the variables in ascending order based on their assigned score.

Fan Li [16] builds on this approach and removes the order requirement without adding additional assumptions. Fan Li [16] applies a Wishart prior to the precision matrix, and uses Lasso regression to perform feature selection. Applying the prior to the precision matrix allows the method to propagate the prior distribution to the regression coefficients, which enables the Lasso regression method to perform Bayesian feature selection. In addition, applying the prior to Ω lets allows Fan Li to prove all possible resulting Bayesian Networks encode the same GGM joint probability, regardless of variable order. This means that the computed Ω is a MAP estimate, which has very appealing properties such as avoiding overfitting. In addition, the [17] 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. However, exploring nonlinear dependencies via this method is future work, because naively implementing the method on our data will produce non-sparse GGM models.

While the method presented by Fan Li [16] 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, 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. While priors are generally reserved for problems with more features than observation samples (e.g., microarray data sets), our E+ data set requires priors as well. Our E+ simulations generate 30,540 output data vectors per input parameter set, which leads to a significant imbalance between output and input observations. For example, 299 E+ simulations produce approximately 3.9GB of data. In addition, the simulation input parameter space represents a complex high dimensional state space, which further complicates the problem. The input parameter space and observational skewness has lead us to avoid exploring direct approaches such as [7], which will most likely only

find a local maximum solution.

Lastly, traditional solvers or optimization techniques are not able to scale to our problem size. We present our solution to this problem in Section 4.1, which allows us to scale to arbitrarily large problems.

4. APPROACH

Given that the regression structure learning method has the most scalability, we only need to address the Lasso regression component's scalability. Once addressed, we can learn the the GGM model using the method outlined in Section 4.2.

Originally, Lasso regression models were learned using quadratic programming methods, which include interior point [20], gradient methods, and many more. However, these traditional methods scaled poorly to large datasets [12]. Improving optimization algorithms' performance and scalability is an active research area, and much advancement has been made over the years. As a result, there are many different methods that are able to scale well to different problem types [6, 24]. For example, the Convex Bundle Cutting Plane (CBCP) method [24] 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. Essentially, the slave computers solve subproblems, and 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) [3] 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. This makes it more efficient to redundantly solve the problem locally, rather than communicate the solution to slave computers.

4.1 Lasso with ADMM

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 subproblems and optimizing over those individual problems with limited communication.

In more formal terms, there exist several common substructures for constrained convex optimization problems [3]. In particular, the general minimization problem is defined as follows:

$$\text{minimize} f(x) \tag{9}$$

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

$$\text{minimize} f(x) + g(z) \tag{10}$$

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 [3] used this general solution format to solve many different convex optimization problems, we are only focusing on the version used to solve Lasso regression. The distributed optimization steps for solving large scale linear Lasso regression problems are presented below[1].

$$x_i^{k+1} = \underset{x_i}{\text{argmin}}(\frac{1}{2}||A_i x_i - b_i||_2^2 + \frac{\rho}{2}||x_i - z^k + u_i^k||_2^2) \tag{11}$$

$$z^{k+1} = S_{\frac{\lambda}{\rho N}}(\bar{x}^{k+1} + \bar{u}^k) \tag{12}$$

$$u_i^{k+1} = u_i^k + x_i^{k+1} - z^{k+1} \tag{13}$$

The individual local subproblems are solved using ridge regression, and the global z values are computed by evaluating a soft thresholding function S. This function is defined as follows:

$$S_{\frac{\lambda}{\rho N}}(v) = max(0, v - \frac{\lambda}{\rho N}) - max(0, -v - \frac{\lambda}{\rho N}) \tag{14}$$

The soft thresholding function 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, computationally demanding, step is solved exactly once. The direct method for computing x_i^{k+1} requires computing the matrix $(A^\intercal A + \rho I)^{-1}$. 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.

4.2 Bayesian Regression GGM Learning

Unlike the direct methods, the Bayesian approach assumes a global Wishart prior for the precision matrix. Using this global prior over the precision matrix allowed Fan Li [16] 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 by Fan Li [16], all variable orderings should theoretically produce the same precision matrix.

The Wishart prior used by Fan Li [16] is defined as $W(\delta, T)$. δ represents a user defined hyperparameter and T is a hyperparameter diagonal matrix who's entries are governed by the following distribution:

$$P(\theta_i) = \frac{\gamma}{2} exp(\frac{-\gamma\theta_i}{2}) \tag{15}$$

[1]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. [3] presents the ADMM formulation for supporting this functionality.

γ is a user defined hyperparameter. Using the above prior and some additional derivations, Fan Li [16] derived the following maximum a posteriori (MAP) distributions for all β and all ψ^{-1}:

$$P(\beta_i|\psi_i, D) \propto exp(\frac{\sum_{n=1}^{D}(x_{ni} - \sum_{j=i+1}^{N}\beta_{ij}x_{nj})^2 + \sqrt{\gamma\Psi_i}\sum_{j=i+1}^{N}|\beta_{ij}|}{-\psi_i}) \quad (16)$$

$$P(\psi_i^{-1}|\theta_i, \beta_i, D) \propto P(D|\Psi_i^{-1}, \beta_i, \theta_i)P(\beta_i|\Psi_i^{-1}, \theta_i)P(\psi_i^{-1}|\theta_i) \sim Gamma(\frac{\delta + 1 + N - 2i + |D|}{2}, \frac{\sum_{j=i+1}^{N}\beta_{ij}^2\theta_i^{-1} + \theta_i^{-1} + \sum_{n=1}^{D}(x_{ni} - \sum_{j=i+1}^{N}\beta_j x_{nj})^2}{2}) \quad (17)$$

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 $\sqrt{\gamma\psi_i}$ [16]. 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[2] used by [16], we solve the Lasso regression problems using ADMM (Section 4.1).

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 Fan Li [16] noted that there is not a known method to analytically remove the hyperparameter via integration. 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,∞), which means Eq. 17 must be transformed to a bounded integral for the numerical methods to be applicable. Given, Eq 17's complex nature and Li's [16] 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 estimates for ψ_i^{-1}, which is the MAP estimate under a Gamma distribution. Based on Fan Li's [16] MAP distribution, we derived the following maximum likelihood estimate (MLE) equation for ψ_i^{-1}:

$$\psi_i^{-1} = \frac{\delta - 1 + N - 2i + |D|}{\sum_{j=i+1}^{N}\beta_{ij}^2\theta_i^{-1} + \theta_i^{-1} + \sum_{n=1}^{D}(x_{ni} - \sum_{j=i+1}^{N}\beta_{ij}x_{nj})^2} \quad (18)$$

Given a fixed θ_i^{-1}, we can estimate a ψ_i^{-1} sample by computing the maximum likelihood estimate (MLE) (Eq. 18). By computing multiple MLE samples according to the θ_i's distribution defined in Eq. 15, we are able to estimate $\mathrm{E}[\psi_i^{-1}|\theta_i, \beta_i, D]$ using weighted sampling. In order to use weight sampled, we sample Eq. 15 using its CDF and a unifrom distribution on the interval [0,1]. This means our final ψ_i^{-1} estimate is computed using the following equation:

$$\psi_i^{-1} = \frac{1}{M}\sum_{j=1}^{M}\hat{\psi_j}^{-1}P(\theta_j) \quad (19)$$

where M is the total number of samples and $\hat{\psi_i}^{-1}$ is a sample computed using Eq. 18. Afterward a few iterations between estimating βs and ψ^{-1}s, we use the final estimates to compute the GGM's precision matrix (Eq. 7).

[2]A gradient based constrained set optimization method.

5. RESULTS

In order to assess how well the GGM estimates building parameters, we experimented with three data sets – Fine Grain (FG), Model Order 1 (MO1), and Model Order 2 (MO2). The first data set contains building simulations built from small brute force increments across 14 building parameters. The MO1 data set contains 299 building simulations and ∼150 building parameters. The building parameters were independently set to their minimum or maximum value, while other parameters were set to their average value. This results in two simulations per building parameter. The MO2 data set contains pairwise minimum and maximum combinations across the same building parameter set.

Using the FG data set, we built a GGM using 250 simulations sampled without replacement, and evaluated the model using 150 test simulations simulations. In addition, we built a GGM using the entire MO1 data set. The resulting model is evaluated using 300 sampled MO2 simulations.

Analyzing Figure 1 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 are not statistically different.

While the other variables are not statistically different, there is not a clear indication that the GGM model fails to represent these variables. This indicates that the GGM overall is successfully modeling the FG building parameters. In fact analyzing Figures 1(a) indicates that the GGM isolates the building parameter's means very well. While the uniform distribution matches the mean and variances (Figure 1(b)), 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.

Similar to the FG data set results, the Bayesian GGM presents excellent performance on the MO2 data set (Figure 2). The Bayesian method is better than a uniform distribution at estimating all building parameters. The Bayesian GGM has better overall error — 13.01±0.85 vs 41.6936±2.13. In addition, the model produces statistically better predictions for all variables.

6. DISCUSSION

While the parameters in FG and MO2 have well defined means, they have instances where they significantly deviate from their estimated means, which is not well implied by results (Figure 1(a) and 2). Figure 3(a) illustrates that

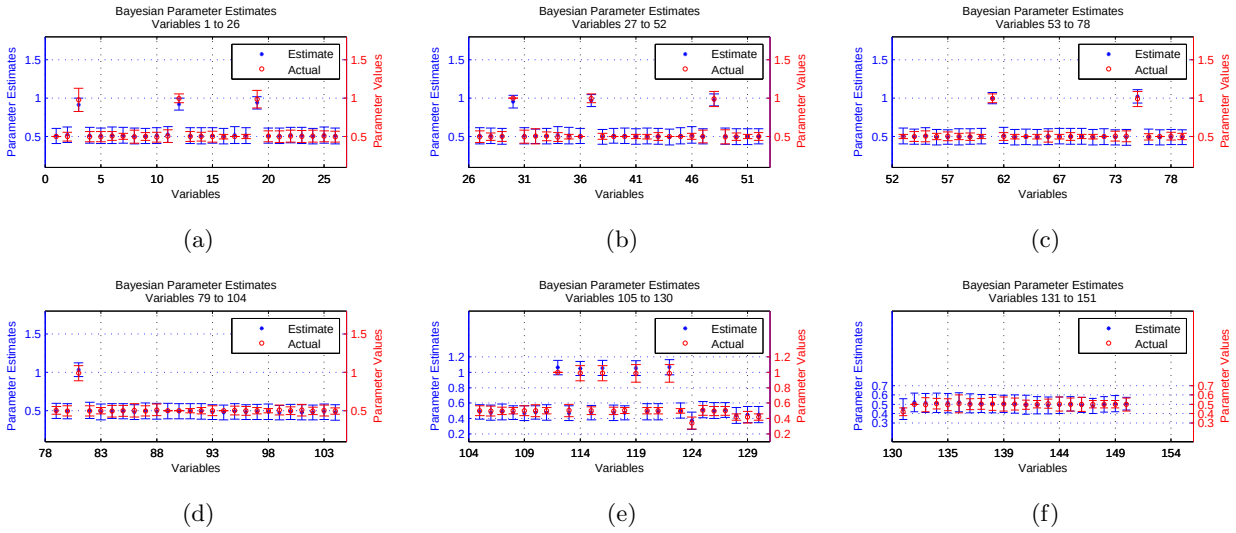
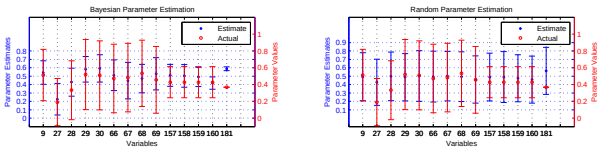
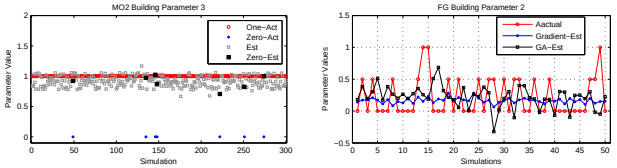


Figure 2: Bayesian GGM's parameter estimates compared against the actual parameter values on 300 randomly sampled MO2 simulations.



(a) Bayesian Estimates (b) Random Estimates

Figure 1: Compares Bayesian GGM's error against a $[0, 1]$ uniform distribution's error on estimating FG building parameters. Additionally, it shows how estimates align with test building parameter values.



(a) (b)

Figure 3: Figure 3(a) highlights that building parameter three has values that occasionally differ greatly from the parameter's mean, and Figure 3(b) compares parameters estimated via a genetic algorithm and gradient optimization.

variable three from Figure 2(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 3(a) 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 MO2 data set present the same behavior.

The FG data set presents similar behavior. Figure 3(b) 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.

7. CONCLUSION & FUTURE WORK

Our FG and MO2 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.

8. ACKNOWLEDGMENTS

This work was funded by field work proposal CEBT105 under the Department of Energy Building Technology Activity Number BT0305000. We would like to thank Amir Roth for his support and review of this project. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. Our work has been enabled and supported by data analysis and visualization experts at the NSF funded RDAV (Remote Data Analysis and Visualization) Center of the University of Tennessee, Knoxville (NSF grant no. ARRA-NSF-OCI-0906324 and NSF-OCI-1136246).

Oak Ridge National Laboratory is managed by UT-Battelle, LLC, for the U.S. Dept. of Energy under contract DE-AC05-00OR22725. This manuscript has been authored by UT-Battelle, LLC, under Contract Number DEAC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

9. REFERENCES

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

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

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

[4] D. Chickering. Learning equivalence classes of bayesian-network structures. *The Journal of Machine Learning Research*, 2:445–498, 2002.

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

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

[7] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[8] N. Friedman. Learning bayesian network structure from massive datasets: The Òsparse candidateÓ algorithm background: Learning structure. *Science*, pages 206–215, 1999.

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

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

[11] T. Hong, F. Buhl, P. Haves, S. Selkowitz, and M. Wetter. Comparing computer run time of building simulation programs. Lawrence Berkeley National Laboratory, 2008.

[12] T. Joachims. Making large-Scale SVM Learning Practical. *Advances in Kernel Methods Support Vector Learning*, pages 169–184, 1999.

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

[14] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. The MIT Press, 2009.

[15] P. Larrañaga, M. Poza, Y. Yurramendi, R. Murga, and C. Kuijpers. 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, 1996.

[16] F. Li. *Structure learning with large sparse undirected graphs and its applications*. PhD thesis, Carnegie Mellon University, 2007.

[17] F. Li, Y. Yang, and E. Xing. From lasso regression to feature vector machine. *Advances in Neural Information Processing Systems*, 18:779, 2006.

[18] D. Margaritis and S. Thrun. Bayesian network induction via local neighborhoods. *Trans. on Pattern Analysis and Machine Intelligence*, 1999.

[19] A. Miller. *Subset selection in regression*, volume 95. CRC Press, 2002.

[20] Y. Nesterov, A. Nemirovskii, and Y. Ye. *Interior-point polynomial algorithms in convex programming*, volume 13. SIAM, 1994.

[21] J. Pellet and A. Elisseeff. Using markov blankets for causal structure learning. *The Journal of Machine Learning Research*, 9:1295–1342, 2008.

[22] G. Schwarz. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[23] P. Spirtes, C. Glymour, and R. Scheines. *Causality from probability*. Carnegie-Mellon University, Laboratory for Computational Linguistics, 1989.

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

[25] I. Tsamardinos, C. Aliferis, and A. Statnikov. 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, 2003.

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

[27] A. Willsky, D. Malioutov, et al. *Approximate inference in Gaussian graphical models*. PhD thesis, Massachusetts Institute of Technology, 2008.