
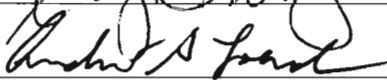


ANALYSIS OF THE RELATIONSHIP BETWEEN PARTIALLY DYNAMIC
BAYESIAN NETWORK ARCHITECTURE AND INFERENCE ALGORITHM
EFFECTIVENESS

by

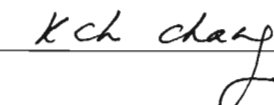
Stephen J. Cannon
A Thesis
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Master of Science
Systems Engineering

Committee:





Dr. Kathryn Laskey, Dissertation Director

Dr. Andrew Loerch, Committee Member



Dr. Kuo-Chu Chang, Committee Member



Dr. Daniel Menascé, Associate Dean for
Research and Graduate Studies



Dr. Lloyd J. Griffiths, Dean, The Volgenau
School of Information Technology and
Engineering

Date: December 4, 2007

~~Summer~~ ^{Fall} Semester 2007
George Mason University
Fairfax, Virginia

Analysis of the Relationship between Partially Dynamic Bayesian Network Architecture
and Inference Algorithm Effectiveness

A thesis submitted in partial fulfillment of the requirement for the degree of Master of
Science at George Mason University

By

Stephen J. Cannon
Bachelor of Science
George Mason University, 2002

Director: Dr. Kathryn Blackmond Laskey, Professor
Department of Systems Engineering and Operations Research

Fall Semester 2007
George Mason University
Fairfax, VA

ACKNOWLEDGEMENTS

This research was supported in part by a grant to George Mason University by the Massachusetts Institute of Technology, Lincoln Laboratory, under contract number PO CX-25963.

TABLE OF CONTENTS

	Page
List of Tables.....	iv
List of Figures	v
Abstract	x
1. Background	1
Problem Statement.....	1
Literature Review	2
Research Objectives and Scope.....	21
2. Methodology	23
Research Factors.....	23
Experimental Methodology.....	42
3. Experimental Results	47
General Analysis Methodology.....	47
Analysis and Results for the Particle Filter Speed of Inference.....	53
Analysis and Results for the Particle Filter Variance of Speed of Inference.....	61
Analysis and Results for the Boyen-Koller Speed of Inference.....	70
Analysis and Results for the Boyen-Koller Variance of Speed of Inference.....	79
Analysis and Results for the Particle Filter Inference Algorithm Accuracy of Inference.....	93
Analysis and Results for the Boyen-Koller Inference Algorithm Accuracy of Inference.....	100
4. Conclusions	105
Discussion of the Time Models.....	105
Discussion of the Accuracy Models	118
Future Work	119
A: Regression Diagnostics for the Particle Filter Speed of Inference Models	123
B: Regression Diagnostics for the Particle Filter Variance of Speed of Inference Models	139
C: Regression Diagnostics for the Boyen-Koller Speed of Inference Models	154
D: Regression Diagnostics for the Boyen-Koller Variance of Speed of Inference Models	166
E: Regression Diagnostics for the Boyen-Koller Variance of Speed of Inference Models	184
List of References.....	190
Curriculum Vitae	192

LIST OF TABLES

Table	Page
1. Factors to be Examined in this Research Study	24
2. Experimental Factors to be Used in this Research Study	27
3. Experimental Factor Distributions.....	28
4. Speed Emphasis Levels for the Boyen-Koller Inference Algorithm	36
5. Speed Emphasis Levels for the Particle Filtering Inference Algorithm	37
6. Speed and Power Statistics on Four Computers used in Study.....	41
7. Assessment Metrics	47
8. Particle Filter Average Speed Exponential Regression Model Statistics	106
9. Particle Filter Average Speed Linear Regression Model Regression Statistics.....	106
10. Particle Filter Average Speed Linear Regression Model Statistics.....	107
11. Particle Filter Average Speed Linear Regression Model Regression Statistics.....	107
12. Particle Filter Variance of the Average Speed Exponential Regression Model Statistics	109
13. Particle Filter Variance of the Average Speed Exponential Regression Model Regression Statistics	109
14. Boyen-Koller Average Speed Exponential Regression Model Statistics	110
15. Particle Filter Average Speed Linear Regression Model Regression Statistics.....	110
16. Boyen-Koller Average Speed Linear Regression Model Statistics.....	111
17. Boyen-Koller Average Speed Linear Regression Model Regression Statistics.....	111
18. Boyen-Koller Variance of the Speed Exponential Regression Model Statistics.....	113
19. Particle Filter Variance of Speed Linear Regression Model Regression Statistics...	113
20. Boyen-Koller Variance of Speed Linear Regression Model Statistics.....	114
21. Boyen-Koller Variance of Speed Linear Regression Model Regression Statistics...	114
22. Boyen-Koller Variance of the Speed Exponential Regression Model Statistics.....	115
23. Particle Filter Variance of Speed Linear Regression Model Regression Statistics...	115
24. PDBN Design Parameters to the Random PDBN Generator.....	188

LIST OF FIGURES

Figures	Page
1. Bayesian Network.....	4
2. Conditional Probability Table of Node ObservedSize.....	5
3. A Bayesian Network with Different Evidence Applied.....	6
4. Partially Dynamic Bayesian Network (6 Time Steps Shown)	12
5. 2-PDBN (Present Time Step and Interface showing)	15
6. Comparison of Results on Various Computers	41
7. Formal Experimentation Flow Chart	44
8. Relationship Between Average Time per Time Step and Variance of Time per Time Step for the Particle Filter Inference Algorithm.....	63
9. Relationship Between Average Time per Time Step and Variance of Time per Time Step for the Boyen-Koller Inference Algorithm.....	80
10. Relationship Between Average Time per Time Step and Variance of Time per Time Step for the Boyen-Koller Inference Algorithm with User Defined Clusters.....	81
11. Relationship Between Average Time per Time Step and Variance of Time per Time Step for the Fully Factored Boyen-Koller Inference Algorithm and the SPI Algorithm	82
12. Particle Filter Average Speed Exponential Regression Model 1 Residuals versus Number of Particles Graph.....	123
13. Particle Filter Average Speed Exponential Regression Model 2 Residuals versus Number of Particles Graph.....	124
14. Particle Filter Average Speed Exponential Regression Model 2 Residuals versus Number of Particles Graph Before Removal of Outliers	124
15. Particle Filter Average Speed Exponential Regression Model 2 Residuals versus Average CPT Size Graph	125
16. Particle Filter Average Speed Exponential Regression Model 3 Residuals versus Number of Particles Graph.....	125
17. Particle Filter Average Speed Exponential Regression Model 4 Residuals versus the Average States per Node Graph	126
18. Particle Filter Average Speed Exponential Regression Model 4 Residuals versus the Natural Logarithm of the Average CPT Size Graph.....	126
19. Particle Filter Average Speed Exponential Regression Model 4 Residuals versus Number of Nodes Graph	127
20. Particle Filter Average Speed Exponential Regression Model 4 Residuals versus the Natural Logarithm of the Number of Particles Graph	127

21.	Particle Filter Average Speed Exponential Regression Model 4 Residuals versus the Natural Logarithm of the Number of Particles Graph Before Removal of Outliers	128
22.	Particle Filter Average Speed Exponential Regression Model 4 Predicted versus Actual Graph	129
23.	Particle Filter Average Speed Exponential Regression Model 4 Predicted versus Actual Graph in Logarithm Space	130
24.	Particle Filter Average Speed Exponential Regression Model 4 Residuals Plot ...	131
25.	Particle Filter Average Speed Exponential Regression Model 4 Q-Q Plot	132
26.	Particle Filter Average Speed Linear Regression Model 1 Residuals versus Average States per Node Graph	133
27.	Particle Filter Average Speed Linear Regression Model 1 Residuals versus Average CPT Size Graph	133
28.	Particle Filter Average Speed Linear Regression Model 1 Residuals versus Number of Dynamic Non-Transitional Nodes Graph	134
29.	Particle Filter Average Speed Linear Regression Model 1 Residuals versus Number of Dynamic Transitional Nodes Graph	134
30.	Particle Filter Average Speed Linear Regression Model 1 Residuals versus Number of Static Nodes Graph	135
31.	Particle Filter Average Speed Linear Regression Model 1 Residuals versus Number of Particles Graph	135
32.	Particle Filter Average Speed Linear Regression Model 1 Predicted versus Actual Graph	136
33.	Particle Filter Average Speed Linear Regression Model 1 Residuals Plot	137
34.	Particle Filter Average Speed Linear Regression Model 1 Q-Q Plot	138
35.	Particle Filter Average Speed Linear Regression Model 2 Residuals versus Number of Nodes Graph	138
36.	Particle Filter Variance of Speed Exponential Regression Model 1 Residuals versus Number of Particles Graph	139
37.	Particle Filter Variance of Speed Exponential Regression Model 2 Residuals versus Number of Particles Graph	139
38.	Particle Filter Variance of Speed Exponential Regression Model 2 Residuals versus Average CPT Size Graph	140
39.	Particle Filter Variance of Speed Exponential Regression Model 3 Residuals versus Average States per Node Graph	140
40.	Particle Filter Variance of Speed Exponential Regression Model 3 Residuals versus the Natural Logarithm of the Average CPT Size Graph	141
41.	Particle Filter Variance of Speed Exponential Regression Model 3 Residuals versus Number of Nodes Graph	141
42.	Particle Filter Variance of Speed Exponential Regression Model 3 Residuals versus the Natural Logarithm of the Number of Particles Graph	142
43.	Particle Filter Variance of Speed Exponential Regression Model 3 Predicted versus Actual Graph	143

44.	Particle Filter Variance of the Speed Exponential Regression Model 3 Predicted versus Actual Graph in Logarithm Space	144
45.	Particle Filter Variance of Speed Exponential Regression Model 3 Residuals Plot	145
46.	Particle Filter Variance of Speed Exponential Regression Model 3 Q-Q Plot	146
47.	Particle Filter Variance of Speed Linear Regression Model 1 Residuals versus Number of Particles Graph.....	146
48.	Particle Filter Variance of Speed Linear Regression Model 2 Residuals versus Number of Particles Graph.....	147
49.	Particle Filter Variance of Speed Linear Regression Model 3 Residuals versus Average States per Node Graph	147
50.	Particle Filter Variance of Speed Linear Regression Model 3 Residuals versus Average CPT Size Graph	148
51.	Particle Filter Variance of Speed Linear Regression Model 3 Residuals versus Number of Dynamic Non-Transitional Nodes Graph.....	148
52.	Particle Filter Variance of Speed Linear Regression Model 3 Residuals versus Number of Dynamic Transitional Nodes Graph.....	149
53.	Particle Filter Variance of Speed Linear Regression Model 3 Residuals versus Number of Static Nodes Graph	149
54.	Particle Filter Variance of Speed Linear Regression Model 3 Residuals versus Number of Particles Squared Graph	150
55.	Particle Filter Variance of Speed Linear Regression Model 3 Predicted versus Actual Graph	151
56.	Particle Filter Variance of Speed Linear Regression Model 3 Residuals Plot.....	152
57.	Particle Filter Variance of Speed Linear Regression Model 3 Q-Q Plot.....	153
58.	Boyen-Koller over SPI Average Speed Exponential Regression Model 1 Residuals versus Average CPT Size Graph	154
59.	Boyen-Koller over SPI Average Speed Exponential Regression Model 2 Residuals versus States per Node Graph.....	154
60.	Boyen-Koller over SPI Average Speed Exponential Regression Model 2 Residuals versus Natural Logarithm of the Average CPT Size Graph.....	155
61.	Boyen-Koller over SPI Average Speed Exponential Regression Model 2 Residuals versus Number of Nodes Graph	155
62.	Boyen-Koller over SPI Average Speed Exponential Regression Model 2 Residuals versus Average Cluster Size Graph	156
63.	Boyen-Koller over SPI Average Speed Exponential Regression Model 2 Predicted versus Actual Graph.....	157
64.	Boyen-Koller over SPI Average Speed Exponential Regression Model 2 Predicted versus Actual Graph in Logarithm Space	158
65.	Boyen-Koller over SPI Average Speed Exponential Regression Model 2 Residuals Plot.....	159
66.	Boyen-Koller over SPI Average Speed Exponential Regression Model 2 Q-Q Plot	160

67.	Boyen-Koller over SPI Average Speed Linear Regression Model 5 Residuals versus Average CPT Size Graph	160
68.	Boyen-Koller over SPI Average Speed Linear Regression Model 5 Residuals versus Number of Dynamic Transitional Nodes Graph.....	161
69.	Boyen-Koller over SPI Average Speed Linear Regression Model 5 Residuals versus Number of Static Nodes Graph	161
70.	Boyen-Koller over SPI Average Speed Linear Regression Model 5 Residuals versus Average Cluster Size Graph	162
71.	Boyen-Koller over SPI Average Speed Linear Regression Model 5 Predicted versus Actual Graph	163
72.	Boyen-Koller over SPI Average Speed Linear Regression Model 5 Residuals Plot	164
73.	Boyen-Koller over SPI Average Speed Linear Regression Model 5 Q-Q Plot.....	165
74.	Boyen-Koller over SPI Variance of Speed Exponential Regression Model 1A Residuals versus Average CPT Size Graph	166
75.	Boyen-Koller over SPI Variance of Speed Exponential Regression Model 1A Residuals versus Average Cluster Size Graph	166
76.	Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2A Residuals versus Average States per Node Graph.....	167
77.	Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2A Residuals versus the Natural Logarithm of the Average CPT Size Graph	167
78.	Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2A Residuals versus Number of Nodes Graph	168
79.	Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2A Residuals versus Average Cluster Size Graph	168
80.	Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2A Predicted versus Actual Graph	169
81.	Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2A Predicted versus Actual Graph in Logarithm Space.....	170
82.	Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2A Residuals Plot	171
83.	Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2A Q-Q Plot	172
84.	Boyen-Koller over SPI Variance of Speed Linear Regression Model 4A Residuals versus Average States per Node Graph.....	172
85.	Boyen-Koller over SPI Variance of Speed Linear Regression Model 4A Residuals versus Number of Dynamic Non-Transitional Nodes Graph.....	173
86.	Boyen-Koller over SPI Variance of Speed Linear Regression Model 4A Residuals versus Number of Dynamic Transitional Nodes Graph.....	173
87.	Boyen-Koller over SPI Variance of Speed Linear Regression Model 4A Residuals versus Number of Static Nodes Graph.....	174
88.	Boyen-Koller over SPI Variance of Speed Linear Regression Model 4A Predicted versus Actual Graph.....	175

89.	Boyen-Koller over SPI Variance of Speed Linear Regression Model 4A Residuals Plot	176
90.	Boyen-Koller over SPI Variance of Speed Linear Regression Model 4A Q-Q Plot	177
91.	Boyen-Koller over SPI Variance of Speed Exponential Regression Model 1B Residuals versus Average CPT Size Graph	177
92.	Boyen-Koller over SPI Variance of Speed Linear Regression Model 2B Residuals versus Average States per Node Graph.....	178
93.	Boyen-Koller over SPI Variance of Speed Linear Regression Model 2B Residuals versus the Natural Logarithm of the Average CPT Size Graph	178
94.	Boyen-Koller over SPI Variance of Speed Linear Regression Model 2B Residuals versus Number of Nodes Graph	179
95.	Boyen-Koller over SPI Variance of Speed Linear Regression Model 2B Residuals versus Average Cluster Size Graph	179
96.	Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2B Predicted versus Actual Graph	180
97.	Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2B Predicted versus Actual Graph in Logarithm Space	181
98.	Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2B Residuals Plot	182
99.	Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2B Q-Q Plot	183
100.	Boyen-Koller over SPI Variance of Speed Exponential Regression Model 1B Residuals versus Average CPT Size Graph	183
101.	Use Case Diagram for the Random PDBN Generator.....	185
102.	Data Flow Diagram for the Random PDBN Generator	186

ABSTRACT

ANALYSIS OF THE RELATIONSHIP BETWEEN PARTIALLY DYNAMIC BAYESIAN NETWORK ARCHITECTURE AND INFERENCE ALGORITHM EFFECTIVENESS

Stephen J. Cannon, M.S.

George Mason University, 2007

Thesis Director: Dr. Kathryn Blackmond Laskey

This thesis examines the relationship between the architecture of partially dynamic Bayesian networks and the effectiveness of various inference algorithms using these Bayesian networks. The algorithms studied were the symbolic probabilistic inference algorithm, the particle filter inference algorithm, and Boyen-Koller inference algorithm. The purpose of this research is to provide empirical support for theoretical models of the speed and accuracy of each of these inference algorithms as well as to develop statistical models that utilize computationally and conceptually simple factors. The author shows that the empirical results for the speed of inference of each inference algorithm generally agrees with the theoretical complexity models of each algorithm. The author also developed empirical models that predict the variance of speed of each of the inference algorithms explored in this research.

1. Background

1.1 Problem Statement

One of the most challenging problems in artificial intelligence is the problem of object classification. Object classification is the problem of inferring the class of an object from information provided by sensor data or other sources. However, the data used for classification may be unreliable and incomplete. The data may not be very informative of the object's class. This same data may be copious and delivered at very high speed. Nonetheless, in real time discrimination, only a limited amount of time may be available for processing the information and coming to a result.

Any combination of these problems makes object classification a very difficult problem. Consider as an example the scenario of a volley of intercontinental ballistic missiles (ICBM) coming towards an American city. An ICBM normally travels at supersonic speeds, and hence, is an incredibly fast target relative to its size. Thus, an ICBM is very difficult to neutralize as a threat; requiring an extremely rapid reaction time to deal with this threat. Most modern ICBMs also have a variety of defense countermeasures to stymie missile defenses further. Information collected by sensors regarding this missile attack may be incomplete and inaccurate due to these countermeasures, environmental conditions at the time of the attack, or simply random

errors. Finally, because not every airborne object is necessarily a threat, a determination must be made as to what is a threat, what is expendable, and what must not be destroyed.

The decisions facing those defending against airborne threats are more complex than whether or not to shoot. Multiple threats must be separated and classified, and then a determination must be made as to how to respond to each threat. There is a variety of possible responses that vary in effectiveness against missiles, and effectiveness against different countermeasures and collateral damage. Also, decisions must be made as to where limited resources should be concentrated and the relative importance of different types of loss. All these decisions must be made in real-time with copious amounts of data, some of which may be inaccurate or incomplete.

1.2 Literature Review

Bayesian networks can be applied to a wide range of problem domains where object classification is a serious issue. Examples of such time-critical real-world applications include FAA flight tracking and management in dense environments, object identification of satellite imagery, and threat identification in missile defense. This research attempts to provide results that can be generalized to any problem domain.

A Bayesian network is a knowledge-based modeling approach, and thus, has the expressional power to model a wide range of problem domains. The knowledge-based approach encompasses such modeling tools as the decision tree and the Bayesian network. These models provide a means to describe the relationship between attributes of the object. Furthermore, some kinds of knowledge-based models, including a class of

Bayesian networks called dynamic Bayesian networks, can describe the dynamics of a temporally evolving system. This larger modeling vocabulary gives the modeler greater power to describe the system and, in turn, has the potential for more accurate classifications of objects because of this extra information in the model. The knowledge-based approaches are also amenable to novel problems that have similarities with existing, well understood problems.

However, setting up the knowledge base for the model is more difficult than simply running a data set through a learning algorithm. Furthermore, this extra power in articulating the system being modeled typically comes at the price of slower generation of useful results in comparison to other modeling approaches (Dunn, Holt, Laskey, Lyons, Takikawa, Tung, 2002). Thus, the knowledge-based approach is not a panacea.

Ultimately, the knowledge-based approach tends to be well suited to problems where some speed reduction can be accepted for an improvement in accuracy and there is a good understanding of the system being modeled. In this situation, one can take advantage of the full power of the knowledge-based approach models. A knowledge-based approach is generally not suitable for problems where there isn't a great deal of understanding of the system being modeled unless that knowledge can be acquired through sophisticated learning methods.

At this time, however, the application of the knowledge-based approach is relatively new to most problem domains. There is little published literature on performance characteristics of the types of models being considered. Hence, there is a need for systematic studies of performance parameters and the effectiveness of the various

knowledge-based methods. Consequently, this was the approach I used in this research study. The knowledge-based modeling approach I have selected is the Bayesian network.

A Bayesian network is an acyclic directed graph in which each node represents the value of an uncertain hypothesis and arcs between the nodes represent dependency relationships. These dependencies can represent cause and effect relationships as well as statistical associations or logical relationships. Figure 1 shows an example of a Bayesian network representing the relationship between a class of object, features of the object, and observations of the features.

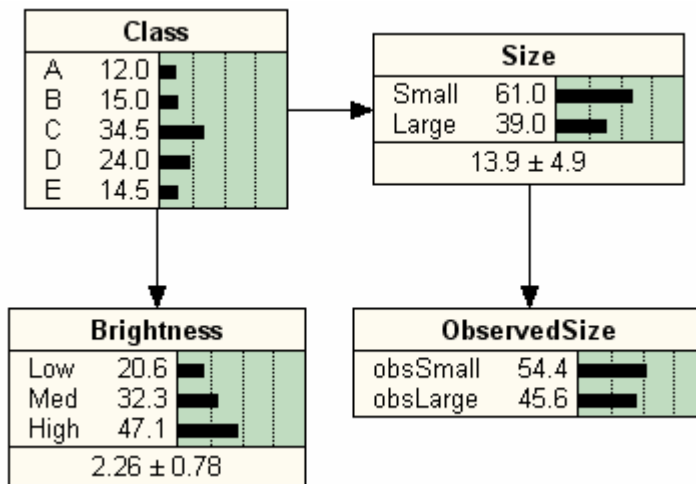


Figure 1: Bayesian Network

In a Bayesian network the direction of the arc depicts the direction of the dependence such that the node receiving the arc is dependent on the node the arc is coming from. For example, the observed size of the object in Figure 1 depends probabilistically on its actual size. This probabilistic relationship is described using a conditional probability table (CPT) like the one shown in Figure 2 that displays the CPT for the node

ObservedSize. In this table, the first column represents the states of the parent node Size and the first row represents the states of the node we are examining, ObservedSize. The probabilities in the table represent the conditional probabilities of the node ObservedSize given the state of the parent node Size. Thus, the probability we observe the size of the object we are examining to be small given that the object is actually large is 30%, as seen in the third row and second column of the table. Nodes that have no parent nodes (such as the node Class in Figure 1) have only one row of probabilities.

Size	obsSmall	obsLarge
Small	70.000	30.000
Large	30.000	70.000

Figure 2: Conditional Probability Table of Node ObservedSize

One can use the Bayesian network in Figure 1 to draw inferences from sensor reports about which type of object is being observed. A conditional probability distribution like the one in Figure 2 is associated with each node in a Bayesian network and defines the probability of each state of the node as a function of the values of its parent nodes (Pearl, 1988; Jensen, 2001).

To infer useful information from any kind of Bayesian network, one must use an inference algorithm. Inference is the process by which one incorporates evidence into a model to produce revised conclusions. An inference algorithm is the algorithm that produces these new conclusions using the model. An inference algorithm for a Bayesian network takes evidence about the values of some nodes and draws new conclusions about the values of other nodes. These new conclusions are in the form of updated distributions for each variable each node represents.

Figure 3 provides an example of this. Here, network A is a Bayesian network where no evidence is applied, network B is a Bayesian network where evidence is applied and a new conclusion is inferred about the remaining nodes, and network C is an alternative set of evidence and inferences. Nodes where evidence has been applied are colored grey while those nodes that new conclusions must be inferred for are off-white. Each network has different findings applied to it, and hence, have inferred different outcomes from that evidence.

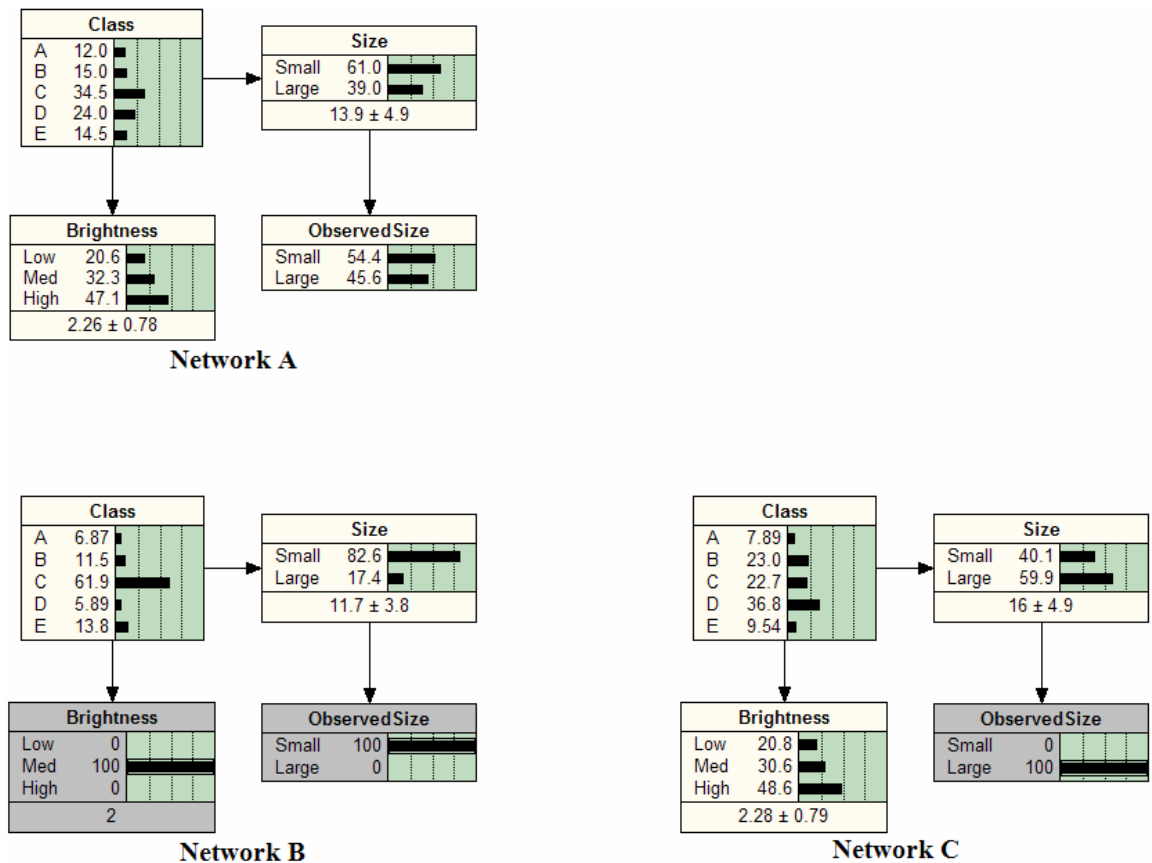


Figure 3: A Bayesian Network with Different Evidence Applied

To understand the resulting probabilities given the different sets of evidence displayed in Figure 3, one must first understand what distinguishes each of the 5 classes

of objects we are trying to identify. Class A, C, and E objects tend to be small while class B and D objects tend to be large. Also, class A objects tend to be very dull (low brightness), class B objects are slightly brighter, class C objects are brighter still, and class D objects are the brightest of all the classes. Class E objects, however, can be any brightness level. Therefore, one cannot easily discriminate class E objects simply by looking at the objects brightness.

Given this information, we can intuitively deduce that a small unidentified object with medium brightness is likely to be a class C object. From the brightness we could expect a class B or D object but they are both large objects. The results of Network B in Figure 3 match our assessment. In this scenario, there is approximately a 62% chance the object given is a class C object.

Simply knowing one piece of information about the object, that the object is large, is not enough information to be certain of the object's classification. However, this observation does lend further credence to the possibility that the given object is either of class B or of class D (both large objects). Comparing the node Class in Network A to Network C in Figure 3, we can see that the probabilities of both class B and class D have gone up. As we can see from these two examples, what our minds are able to do with intuition regarding the classification of objects, Bayesian networks are able to do quantitatively using conditional probability tables.

There are two classes of inference algorithms, exact and approximate. Exact inference algorithms are guaranteed to provide the exact conditional distribution of any node in the Bayesian network given current observations. Approximate inference

algorithms, on the other hand, only approximate the distribution for any node in the Bayesian network. Common approximation methods include structure simplification (i.e., ignoring some dependencies) or Monte Carlo simulation. The advantage of approximate inference algorithms is that they are faster than exact inference algorithms when performing inference over large and complex Bayesian networks.

The inference algorithms examined in this research study are Particle Filtering (Ng, Peshkin, Pfeffer, 2002), the Boyen-Koller algorithm (Boyen, Koller, 1999), and the dynamic symbolic probabilistic inference (SPI) algorithm (D'Ambrosio, Li, 1994, Takikawa, D'Ambrosio, Wright, 2002). Both the Particle Filtering and Boyen-Koller inference algorithms are approximate algorithms, and hence, were added to the study to examine inference algorithms that satisfy the speed objective at the cost of accuracy. The Particle Filtering algorithm uses Monte Carlo simulation while the Boyen-Koller algorithm uses structure simplification. The SPI algorithm, however, is an exact inference algorithm, and thus, examines the opposite side of the spectrum; inference algorithms that satisfy the accuracy objective at the cost of speed.

The SPI method of inference increases the speed of the computation by reducing the number of calculations required to do inference. The SPI method approaches the problem of inference as the computation of sums of products. Examining the problem of inference from this perspective, the algorithm is then able to use the distributive law to organize these sums of products in a way that is computationally efficient when performing inference.

To understand how the SPI algorithm might achieve this, one must understand that the probability distribution of any variable or set of variables given some evidence is the joint probability distribution of the target variables and the evidence variables divided by the joint probability distribution of the evidence variables:

$$P(T_1, \dots, T_n \mid E_1, \dots, E_m) = \frac{P(T_1, \dots, T_n, E_1, \dots, E_m)}{P(E_1, \dots, E_m)} \quad \text{Equation 1}$$

The joint distribution over all the variables in the Bayesian network is simply the product of all the conditional probabilities in the Bayesian network. This property of Bayesian networks is known as the chain rule (Jensen, 2001). Using the chain rule, the two massive probabilities shown on the right side of Equation 1 become a series of small probabilities that must all be multiplied together.

One can then remove variables from the joint distribution, thus simplifying the computation of inference, by marginalizing the joint probability distribution over the variables one wishes to remove. Marginalizing is an operation by which one sums or integrates over the variables to be removed from the joint probability. The advantage of marginalizing variables is that one can remove variables from the inference equation that do not influence the inference query. Marginalization also serves to organize the inference equation into an equation structure that has more products of sums than sums of products in the equation. This structure facilitates more efficient computation. The following is a formula for marginalizing a joint distribution $P(W, X, Y, Z)$ over the variables Y and Z to obtain the marginal distribution $P(W, X)$:

$$\sum_{Y,Z} P(W, X, Y, Z) = P(W, X)$$

Equation 2

The joint probability over all the variables in the Bayesian network could in principle be obtained by computing the numerator and the denominator of Equation 1, which describes how to get the joint probability of a set of target nodes given the evidence. However, this is tractable only for very small networks.

The calculations required to compute the probability of interest given evidence, therefore, consist of a series of multiplication and addition operations. The complexity of this computation depends on the order in which one performs these operations. Ideally, it is better to calculate the product of sums rather than the sum of products whenever possible. The approach of the symbolic probabilistic algorithm is to find a way of arranging these factors to achieve this goal when calculating a conditional probability (D'Ambrosio, Li, 1994).

The Bayesian network of Figure 1 represents a static snapshot of the object being examined. Symbolic probabilistic inference algorithms are designed to handle this kind of static Bayesian network that models static situations. However, many problem domains have variables that are dynamic. These algorithms are not suitable to handle dynamic system without modification. Furthermore, both the particle filtering and the Boyen-Koller inference algorithms are exclusively for use with dynamic systems. For situations where a changing system is observed repeatedly over time, a partially dynamic Bayesian network (PDBN) is required to model the time evolution of the object's features and observations.

A partially dynamic Bayesian network is used to model problems in which the system being modeled changes over time. Variables of the modeled system whose value remains constant are called static variables, and are represented as ordinary Bayesian network nodes. Variables that change with time are called dynamic variables. A PDBN contains multiple copies, or instances, of each dynamic node; one for each time step. Each instance of the dynamic node represents the value of the dynamic variable at that time step. Figure 4 gives an example of a partially dynamic Bayesian network. This example is a dynamic version of the Bayesian network in Figure 1.

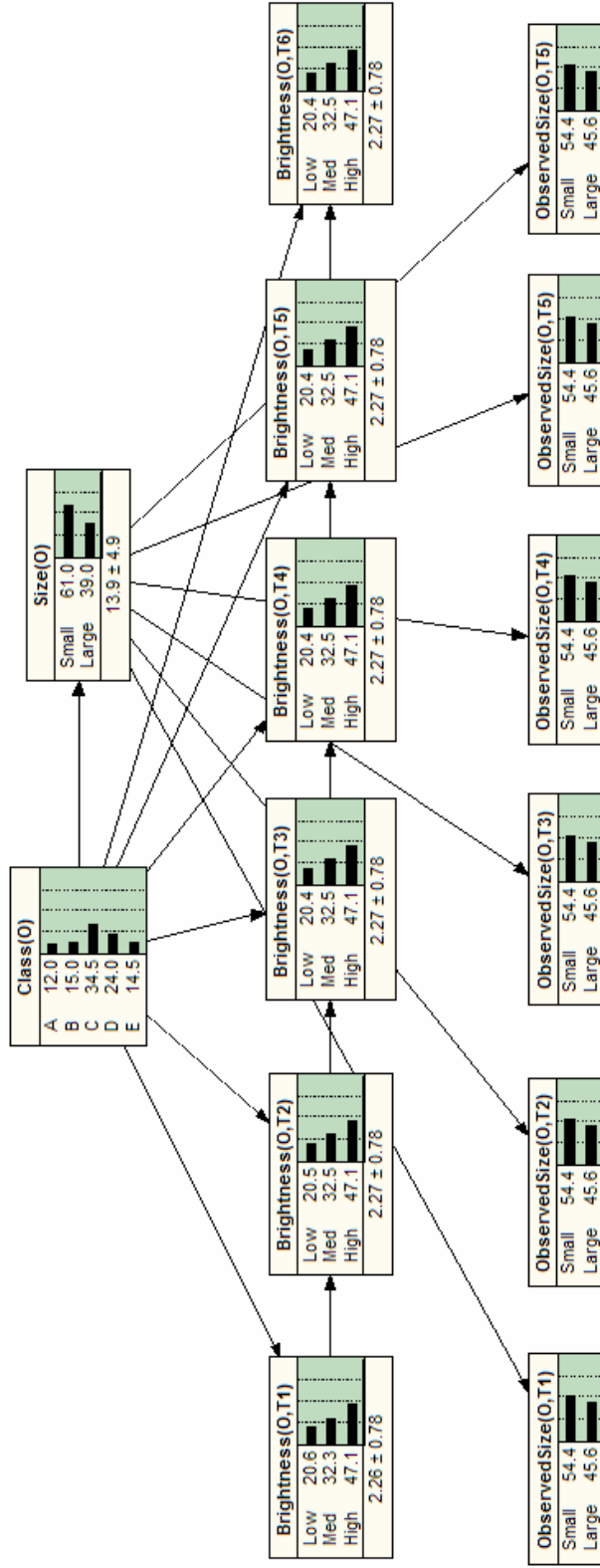


Figure 4: Partially Dynamic Bayesian Network (6 Time Steps Shown)

In Figure 4, brightness and observed size are two variables that are represented by dynamic nodes. Dynamic nodes can be separated further into two groups. Transitional nodes, such as those nodes representing the brightness of the object, are those nodes that have arcs that are directed into the next time step. Other dynamic nodes, such as those nodes in Figure 4 representing the observed size of the object, are referred to as non-transitional dynamic nodes.

There are rules that dictate how one can connect nodes in a PDBN. Static nodes cannot have dynamic parents. The reasoning behind this rule is that for one to consider a variable static, that variable cannot be a function of a dynamic variable within the scope of the system being modeled. If a variable were directly influenced by another variable that changed over time, that variable would have to be dynamic. Dynamic nodes, however, can have either static or dynamic parents.

Together, the static nodes that have dynamic children and the transitional nodes constitute what is called the interface. It is these nodes that influence the next time step. The Bayesian network in Figure 4 is an order one partially dynamic Bayesian network. It is partially dynamic because there exists some static nodes in the network. It is referred to as order one because the furthest extent into the future a node directly influences another node is one time step. An order N PDBN is a PDBN where the furthest extent into the future a node directly influences another node is N time steps. An order N PDBN (N being a variable integer) is also an order N Markov chain with stationary transition probabilities, where the state space of the Markov chain is the cross product of

the state spaces of the static nodes and the dynamic nodes at a given time step (Takikawa et al., 2002).

Partially dynamic Bayesian networks are useful for problems in which observational evidence is received over time about a moving object or set of objects. Static nodes represent intrinsic unchanging properties of the object or objects (e.g., object type; material composition; length). Time-varying properties of the object (e.g., velocity; position; orientation with respect to the sensor) are represented by transitional or non-transitional dynamic nodes. Sensor reports are also represented as dynamic nodes.

Dynamic Bayesian networks and partially dynamic Bayesian networks represent a particularly difficult class of Bayesian networks. Recall that in a dynamic Bayesian network (DBN) or a partially dynamic Bayesian network, there is a copy of each dynamic node for each time step. If the system is followed for many time steps, this results in an extremely large network.

The temporal rollup method was developed to make PDBN inference more efficient by exploiting the Markov property of PDBNs in inference. The Temporal rollup method takes advantage of the fact that dynamic nodes of the present time step are related to nodes several time steps ago only through the interface nodes of the immediately preceding time step (Takikawa et al., 2002). For example, the “ObservedSize” variable in the fourth time step in Figure 4 is only related to the “Brightness” variable in the first time step through the “Brightness” variable in the fourth time step. This property allows the information from all previous time steps to be summarized or rolled up through the interface nodes as shown in Figure 5. Figure 5 displays the 2-PDBN representation of the

partially dynamic Bayesian network in Figure 4. Here, the interface nodes that separate the dynamic nodes in the present time step from all nodes in previous time steps are the static node “Class” and the dynamic transitional node “Brightness_Prev”. The joint distribution over the interface variables is collectively called the past expression. The past expression contains all the information from the previous time steps relevant to future predictions. The inference algorithm uses the past expression to infer the probabilities for nodes in the current time step. Hence, instead of explicitly representing each node in every previous time step, all the information represented in the past time steps are rolled up into a joint probability distribution called the past expression.

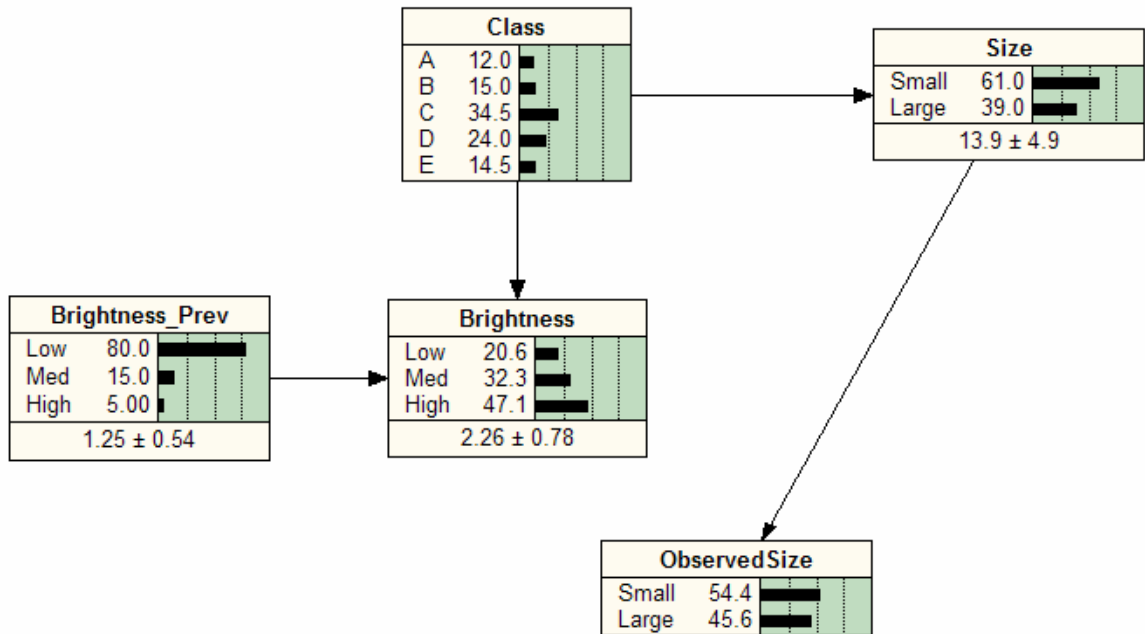


Figure 5: 2-PDBN (Present Time Step and Interface showing)

Thus, if we are interested only in static nodes and the current values of dynamic nodes, then only the current time steps and the interface nodes from the previous time

step need to be explicitly represented for a first order PDBN. N order PDBNs require the explicit representation of the current time step and the past N time steps. The PDBN in Figure 5 is a first order PDBN, and therefore, only requires explicit representation of the current dynamic nodes, static nodes, and the dynamic transitional previous time step nodes.

At each step of the temporal rollup procedure, the current nodes are rolled up with the past interface nodes to form a new interface, and the network is then rolled forward so that the next time step becomes the new current time step. Hence, given a network such as the one in Figure 5, one applies evidence to nodes in the present time steps such as “ObservedSize” and “Brightness”. This new evidence will change the probability distributions of other nodes such as the nodes “Size” and “Class” in Figure 5. To infer the new distribution of a node such as “Size” that is directly influenced by an interface node, one uses the past expression for this time step. This is done by marginalizing this joint probability distribution to determine the joint probability of all the variables that directly influence the variable in question. In this case the past expression consists of “Class” and “Brightness_Prev” with only the static node “Class” directly influencing the node “Size”. In this situation one would marginalize the past expression to obtain only the distribution of the variable “Class”. Once evidence has been applied and the new distributions of the remaining nodes inferred, the rollup procedure can begin. The nodes that will form the interface in the next time step, the static node “Class” and the dynamic transitional node “Brightness”, will be combined into a joint probability distribution. This new past expression will contain all the information gathered in the past time steps

as well as the information collected in this time step. The network for the next time step will then contain the dynamic transitional nodes of this time step and any static nodes and any dynamic nodes in the next time step. In this way, the only nodes that need to be explicitly represented each time step are the static nodes, the dynamic nodes in the present time step, and the dynamic nodes from the previous time steps that directly influence nodes in the present time step (Takikawa et al, 2002).

The temporal rollup procedure is the basis of all the inference algorithms considered in this study. I used the temporal rollup procedure in conjunction with the symbolic probabilistic inference algorithm by applying the algorithm to the rolled-up version of the dynamic Bayesian network. Applying an efficient exact inference algorithm such as the SPI algorithm allows the modeling of dynamic systems using Bayesian networks to be tractable for PDBNs of a limited size and complexity. However, for real-time inference in most problem domains, approximate inference algorithms must be used. This is because the size of the past expression when using the temporal rollup procedure scales as the product of the number of states of each node in the interface. Hence, the number of elements in the past expression of the PDBN represented in Figure 5 will be the product of five states from the “Class” node and three states from the “Brightness_Prev” node or 15 elements. Networks with many more nodes in the interface or nodes with many more states will result in a combinatorial explosion in the size of the past expression, and hence, make inference intractable using exact inference methods on large and complex networks.

One approximation to the temporal rollup procedure is particle filtering. Particle filtering is an approximate inference algorithm that makes use of the Monte Carlo method. Rather than use the full joint distribution on transition nodes from the previous time step, a sample set of node-value pairs, or particles, is selected. Each particle represents an instance of a set of values of all the nodes in the interface. Collectively, the set of particles approximates the joint probability distribution over the nodes in the interface at a given time step conditional on all evidence received to that point. As more particles are used, accuracy improves, but at the cost of more computation. The sample of particles is used in the rollup into the next time step rather than the full joint distribution. The particles are then weighted based on the evidence at the next time step, where weights are proportional to the likelihood of the evidence given the particles. The set of nodes to be used as particles in the next time step is then resampled based on the likelihood weights just collected. This is repeated every time step to speed up rollup of the previous time step into the interface of the next time step (Ng et al., 2002).

The Boyen-Koller inference algorithm is also an approximate inference algorithm. However, rather than utilizing a Monte Carlo approximation, this method approximates the joint distribution on the interface nodes by a tractable distribution. The interface nodes at a given time step are grouped into clusters that are independent or interact weakly in the approximate distribution. Clusters are chosen heuristically in an attempt to capture important dependencies while keeping the two-stage network tractable. As a general guideline, large clusters tend to slow inference speed but increase accuracy while small clusters increase speed at the cost of accuracy. To compute the probability

distribution for the next time step, a two-stage network is used in which the distribution at the current time step is approximated by a product of factors defined on the clusters and the original Bayesian network distributions used for the transition between time steps. The belief tables for the clusters of the next time step are then derived through marginalization (Boyen et al., 1998). With this simplified dynamic Bayesian network, inference over the two-step network can be done using an exact inference algorithm such as the symbolic probabilistic inference algorithm or the junction tree inference algorithm.

The Boyen-Koller inference algorithm has a running time of $O(NQ^{\sqrt{N}})$ per time step where N is the number of nodes in the Bayesian network and Q is the number of states in each node of the network (Murphy et al., 2001). This study done by Murphy and Weiss assumes all nodes have the same number of states, while the PDBNs used in this research study will generally have nodes with different numbers of states. Nonetheless, this expression of computational complexity is still useful in predicting the computation time for the Boyen-Koller algorithm.

The particle filter inference algorithm has a running time of $O(M)$ per time step where M is the number of particles used by the particle filter (Carpenter et al., 1999). This is not to say that the computation time of the particle filter is not a function of the size of the network. The running time of this inference algorithm would likely be $O(MN)$ if one were to include the Number of Nodes factor (here the variable N). However, the number of nodes is generally insignificant in number relative to the number of particles. Thus, the number of particles is considered the more significant factor of the two in the complexity of the particle filter algorithm.

The speed at which the particle filter inference algorithm can run appears to be less sensitive to the size and complexity of the network than the Boyen-Koller inference algorithm. The time to perform inference using the Boyen-Koller inference algorithm is a function of the number of states per node in the PDBN and is especially sensitive to the number of nodes in the PDBN. Both of these factors are different metrics with which to measure the size and complexity of a Bayesian network. Conversely, the time to perform inference using the particle filter inference algorithm is a function of a factor that is not an attribute of the Bayesian network.

However, the number of particles used to perform inference on any PDBN, even a relatively simple network, is generally in the thousands to tens of thousands. It is necessary that this many particles be used to ensure that these particles portray an accurate representative sample of the state of the PDBN. For PDBNs with few nodes and few states per node, the Boyen-Koller inference algorithm is generally the faster algorithm because the particle filter inference algorithm requires a certain amount of overhead to be as accurate an approximate inference algorithm as the Boyen-Koller inference algorithm over the same network.

Thus, the Boyen-Koller inference algorithm will generally be the faster algorithm up to a certain limit of size and complexity of PDBNs. Once that limit is surpassed, the Boyen-Koller inference algorithm will fall behind the particle filter inference algorithm in speed. As the size and complexity of the PDBNs grows linearly, the time to perform inference with the Boyen-Koller inference algorithm will grow exponentially. However,

the time to perform inference with the particle filter inference algorithm will only grow linearly as the number of particles necessary to perform inference accurately grows.

1.3 Research Objectives and Scope

Exact inference algorithms are feasible for simple and small partially dynamic Bayesian networks. However, exact inference is NP-hard (computationally expensive to perform) and as the PDBN grows in size and complexity the time to perform exact inference grows exponentially (Cooper 1990). The computational cost of approximate inference algorithms also grows with problem size, at different rates for different algorithms. The structure and conditional probability distributions of a Bayesian network are important to algorithm effectiveness. However, the performance of an algorithm for a given Bayesian network is usually not easy to determine a priori. Therefore, the objective of this research study is to understand the relationship between the structure and conditional probability tables of PDBNs (what shall be referred to henceforth as the architecture) and the effectiveness of various inference algorithms. An effectively performing algorithm is one that is fast enough to run in real time and accurate enough to provide dependable results and do this within a reasonable resource load (IET, 2002).

This study will be limited to partially dynamic Bayesian networks. Also, the variables of all PDBNs will be restricted to a discretization of a particular functional form I have called the sine normal linear. A sine normal linear variable is one where the variable in question is the sine of a weighted sum of variables. In this particular implementation, every variable in the Bayesian network will be the sine of the weighted

sum of the variables influencing the variable in question. Variables that are root nodes will be the sin of a normal distribution. Every variable being represented in the partially dynamic Bayesian networks being studied will be sine normal linear variables. This particular functional form has been chosen because of its simple implementation and use in experimentation. This function form could be varied in future research, however, and the software tools being used in this research study support such a change.

2. Methodology

2.1 Research Factors

There are many factors of the architecture of PDBNs that may potentially influence the effectiveness of one of the inference algorithms studied. The purpose of this experiment is to identify characteristics that are strongly related to an algorithm's effectiveness and to evaluate the nature of these relationships for the strongly related factors. Several different categories of factors can ultimately affect an inference algorithm's performance. The categories of factors varied in this experiment are: factors describing the PDBN's architecture, the inference algorithms used and metaparameters for those algorithms, the number of time steps the inference algorithm is allowed to run, and the power of the computer running the inference algorithm. One must vary each of these in a controlled manner to be able to analyze with any great clarity what sorts of Bayesian networks work best with what inference algorithms. Of all these factors, only the factors describing the PDBN's architecture are within the scope of this study, and thus, the only category of factors analyzed in detail and compared to the resulting algorithm's effectiveness.

To achieve the objectives of this research study, it was necessary to apply the algorithms being tested to a collection of PDBNs that varied with respect to the attributes hypothesized to affect algorithm effectiveness; the factors that describe the PDBN's

architecture. To achieve this end, I described each PDBN using a set of categorization parameters. Each categorization parameter depicts an attribute of a partially dynamic Bayesian network that I hypothesized as having a high likelihood of affecting the performance or accuracy of one or more of the inference algorithms tested. These categorization parameters were the factors that I tested in this research study.

Table 1 lists a number of factors that can potentially affect the performance or accuracy of the Bayesian network inference algorithms. I drew these factors from the Information Extraction & Transport, Inc.'s Test Plan for Evaluating Approximate Inference Algorithms (IET, 2002) and the expert judgment of Dr. Kathryn Laskey, advisor to this study. The first column in this table numbers the factor labeled in the second column. The third column describes the factor.

Table 1: Factors to be Examined in this Research Study

	Design Factors	Description
1	Number of static nodes	The number of static nodes in the Bayesian network.
2	Number of dynamic transitional nodes	The number of transitional dynamic nodes in the Bayesian network.
3	Number of dynamic non-transitional nodes	The number of non-transitional dynamic nodes in the Bayesian network.
4	Arc saturation for static to static arcs	The percentage of the possible set of arcs between static nodes in the PDBN. 0% represents the minimum number of arcs and 100% represents the maximum number of arcs possible.
5	Average number of static parents for static nodes	The mean number of static parents for the set of static nodes in the PDBN.
6	Arc saturation for static to dynamic arcs	The percentage of the possible set of arcs between static nodes and dynamic nodes in the PDBN. 0% represents the minimum number of arcs and 100% represents the maximum number of arcs possible.
7	Average number of static parents for dynamic nodes	The mean number of static parents for the set of dynamic nodes in the PDBN.

8	Arc saturation for transitional nodes between time steps	The percentage of the possible set of arcs between transitional nodes between time steps in the PDBN. 0% represents the minimum number of arcs and 100% represents the maximum number of arcs possible.
9	Average number of previous time step transitional parents for dynamic nodes	The mean number of transitional parents in the previous time step for the set of dynamic nodes in this time step.
10	Arc saturation for dynamic to dynamic arcs	The percentage of the possible set of arcs between dynamic nodes in the PDBN. 0% represents the minimum number of arcs and 100% represents the maximum number of arcs possible.
11	Average number of same time step dynamic parents for dynamic nodes	The mean number of dynamic parents in the same time step for the set of dynamic nodes in this time step.
12	Proportion of dynamic transitional nodes that have a transitional link to itself	The number of dynamic transitional nodes that link to itself in the next time step divided by the total number of transitional dynamic nodes.
13	Average strength of dependency	Approximate average correlation of nodes connected by an arc.
14	Average node variance	Approximate average variance of every node in the Bayesian network.
15	Average number of states per node	The mean number of states per node.
16	Number of nodes within the interface between time steps	The number of nodes in the interface.
17	Ratio of arcs to nodes	The number of arcs divided by the number of nodes.
18	Maximum number of parents of the nodes	The number of parents of the node with the most parents in the PDBN.
19	Average number of parents of the nodes	The mean number of parents of the nodes in the PDBN.
20	Average size of the CPTs	The mean number of elements of a node's conditional probability table in the PDBN.
21	Total number of possible instances of the PDBN	The total number of possible combinations of states that all the nodes in the PDBN can take on.

An experimenter generating a PDBN can control or influence each of these factors.

However, one cannot control all combinations of these factors simultaneously. For

example, although it is easy to control the number of nodes, the ratio of arcs to nodes, and

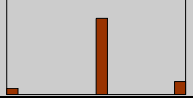
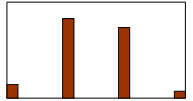
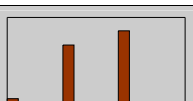


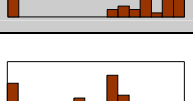
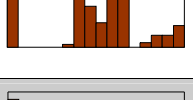
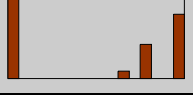
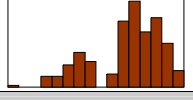
the average number of parents per node in a PDBN, controlling all three of these factors simultaneously is a more complicated task since each of these factors depends on the other two.

Table 2 lists a set of experimental factors I manipulated in this study. The first column in this table numbers the factor labeled in the second column and the third column describes the factor. Table 3 lists these same experimental factors along with various statistics and graphs showing the distributions of each of these factors. The first two columns are identical to those in Table 2. The third column defines the probability density function used for the particular factor when generating PDBNs. The purpose of Table 3 is to display the distribution of these experimental factors in the PDBN data set used in this research study. The random PDBN generator randomly chose a value using these distributions to select the experimental factor values for each PDBN generated. The fourth column gives the full range of values found empirically in a preliminary test run of the random PDBN generator. In this preliminary test run, the random PDBN generator used the distributions found in column three of this table and generated approximately 250 PDBNs for the purpose of collecting the statistics found in Table 3. The fourth column also gives the empirical mean, median, and mode (where applicable) of the resulting set of results in the preliminary test run. The fifth and final column displays a small histogram of the resulting set of results for each factor.

Table 2: Experimental Factors to be Used in this Research Study

	Design Factors	Description
1	Number of static nodes	The number of static nodes in the Bayesian network.
2	Number of dynamic transitional nodes	The number of transitional dynamic nodes in the Bayesian network.
3	Number of dynamic non-transitional nodes	The number of non-transitional dynamic nodes in the Bayesian network.
4	Arc saturation for static to static arcs	The percentage of the possible set of arcs between static nodes in the PDBN. 0% represents the minimum number of arcs and 100% represents the maximum number of arcs possible.
5	Arc saturation for static to dynamic arcs	The percentage of the possible set of arcs between static nodes and dynamic nodes in the PDBN. 0% represents the minimum number of arcs and 100% represents the maximum number of arcs possible.
6	Arc saturation for transitional nodes between time steps	The percentage of the possible set of arcs between transitional nodes between time steps in the PDBN. 0% represents the minimum number of arcs and 100% represents the maximum number of arcs possible.
7	Arc saturation for dynamic to dynamic arcs	The percentage of the possible set of arcs between dynamic nodes in the PDBN. 0% represents the minimum number of arcs and 100% represents the maximum number of arcs possible.
8	Average strength of dependency	Average correlation of all nodes connected by an arc in the PDBN at the initial time step.
9	Average number of states per node	The mean number of states per node.

Table 3: Experimental Factor Distributions

	Design Factors	Factor Distribution	Empirical Range	Empirical Distribution
			Mean/Median/Mode	
1	Number of static nodes	gamma(73.46938776, 0.040833333)	2 – 4 3.072 / 3 / 3	
2	Number of dynamic transitional nodes	gamma(56.25, 0.08)	3 – 6 4.414 / 4 / 4	
3	Number of dynamic non-transitional nodes	gamma(56.25, 0.08)	3 – 6 4.482 / 5 / 5	
4	Arc saturation for static to static arcs	beta(40.25, 189.75)	0 – 0.333 0.041 / 0 / 0	
5	Arc saturation for static to dynamic arcs	beta(35.26, 1727.74)	0 – 0.043 0.022 / 0.031 / 0	
6	Arc saturation for transitional nodes between time steps	beta(34.87063116, 1010.725506)	0 – 0.057 0.027 / 0.029 / 0.031	
7	Arc saturation for dynamic to dynamic arcs	beta(35.26, 1727.74)	0 – 0.036 0.0178 / 0.028 / 0	
8	Average strength of dependency	Unknown PDF StrOfDep > 0.1	-0.743 – 0.681 0.092 / 0.252 / NA	
9	Average number of states per node	beta(7.477777778, 17.44814815) range: (2, 12)	3.824 – 5.188 4.468 / 4.467 / 4.235	

The Average Strength of Dependency experimental factor is unique among the 9 experimental factors for several reasons. This factor could not be controlled directly, and therefore, was controlled indirectly by measuring the average strength of dependency for the produced PDBN and regenerating it if the resulting measure was not greater than 0.1.

It was necessary to enforce a lower bound to the Average Strength of Dependency experimental factor so that the arcs of the PDBNs would be meaningful. The arcs of any Bayesian network represent a relationship between two nodes, and thus, it is taken for granted that two connected nodes will influence each other. Therefore, if one applies evidence to one node one can expect all nodes connected to that node by an arc will be affected. However, one could make a valid and internally consistent Bayesian network in which any or all pairs of interconnected nodes have little or no influence on each other. Such networks are of little practical use and are never purposely created by a knowledge engineer. Thus, to prevent the possibility of the random PDBN generator introducing this sort of Bayesian networks into the sample set, a minimum average strength of dependency was enforced.

I choose the specific value of 0.1 through trial and error. I choose a minimum allowed average strength of dependency value arbitrarily and then examined the sample set of PDBNs to determine whether they were suitable. I tested a small random sample of PDBNs from the sample set to see whether evidence applied to some of the dynamic non-transitional nodes had an effect on the static nodes. If there was an effect on the static nodes, the PDBN was suitable. If all the selected PDBNs from the sample set were suitable, the sample set was suitable. My objective was to find the lowest minimum average strength of dependency value that produced a suitable sample set of PDBNs.

It is important to note that the average strength of dependency measure was approximated using a Monte Carlo simulation that produced results that differed due to random sampling error for each measurement made. The results varied by about 0.01 standard deviations for a single PDBN. This is why some of the PDBNs in the sample set have a reported average strength of dependency as low as 0.083, which is below the 0.1 minimum allowed value.

It is likely that some of the PDBNs generated had an average strength of dependency near 0.1. When the PDBN was tested for suitability, the measured average strength of dependency for the PDBN was above 0.1. However, when that same PDBN was later examined for the purposes of taxonomy, a new average strength of dependency measure was taken. Since the average strength of dependency measure for the PDBN was near 0.1 and each measure has a random sampling error associated with it, this second measure could potentially end up below the 0.1 minimum allowed value.

It was necessary to measure the Average Strength of Dependency experimental factor using a random sampling method so that the measurements could be guaranteed to be computationally tractable. Calculating the average correlation between all the connected nodes of a Bayesian network is an NP-hard problem. The reason for this is that the problem of determining the correlation of two nodes is at its core an inference problem. To determine the correlation of two nodes, one is ultimately finding the expected values of each of the nodes being examined as well as the expected value of the joint of those two nodes. Finding these expected values is an NP-hard problem (Cooper, 1990). Developing a means to calculate directly the average strength of dependency of a PDBN is also a difficult problem beyond the scope of this research project. There is also no guarantee that any means of calculating the average strength of dependency would be computationally tractable for all the PDBNs in my sample set. Using Monte Carlo methods, thus, allowed me to approximate the average strength of dependency of all the PDBNs in my sample set within a reasonable amount of time and with minimal developmental effort.

There exists some bias in the results reported in Table 3. This is why some of the empirical distributions in column 5 of Table 3 do not match up with the theoretical distributions in column 3 of Table 3 as well as one would expect. There are several reasons for the bias. The first source of bias was due to the method of indirectly controlling the average strength of dependency of the PDBNs generated. Since any PDBN generated with an average strength of dependency below 0.1 was not used in the

sample set, it stands to reason that experimental factors that are related to the Average Strength of Dependency factor would become biased.

Another source of bias was from the fact that a handful of PDBNs generated were too complex to perform inference on them, and thus, results could not be generated for these PDBNs. I carefully choose and modified the parameters of each probability distribution for each experimental factor to maximize the number of PDBNs that would be testable while still covering as wide a range of PDBN complexities as possible. However, due to the large number of PDBNs generated and tested, there were still some PDBNs generated that were too complex for the software to handle. Should a PDBN cause the inference algorithm software to exceed its allotted resources or if the test run is unable to be completed in under 10 hours, the test run is forcibly stopped by the experimental test software and all results for that PDBN would be lost. For this reason, experimental factors that contribute to a PDBNs complexity, and ultimately, the difficulty in performing inference on the PDBN will become biased.

It is important to note that these biases did not interfere with the results of this research. The reason for this is because I was able to collect data for each experimental factor for each sample PDBN used in this study. Thus, I was able to fully account for all the biases that existed in the experimental factors.

I choose these ten factors because they are related to the factors listed in Table 1 and hypothesized to influence algorithm effectiveness; e.g., the number of nodes in the interface is related to the number of static nodes, the number of dynamic transitional nodes, and the average number of static parents for dynamic nodes. Furthermore, one

can freely manipulate any combination of these factors when generating a Bayesian network. That is, no combination of factors directly constrains the value of another factor. Finally, it is a reasonable hypothesis that factors dealing with the standard deviation of a PDBN property will have a major influence on the effectiveness of an inference algorithm only if the corresponding average factor has an effect (Laskey, personal communication). For this reason, standard deviations were not included as experimental factors at this time.

I choose the parameters of the probability distributions for each experimental factor primarily based on experience with the various inference algorithms used in this experiment. I had some experience running the inference algorithms using a variety of different PDBNs, and thus, gained a good understanding of what situations would cause the inference algorithms to fail to produce inference results before running out of system resources on a standard personal computer. I applied this knowledge when creating an initial set of parameters for the probability distributions of each experimental factor. When selecting the probability distribution parameters, I also kept in mind what kind of PDBNs would generally be usable in any problem domain. I then proceeded to generate and test PDBNs in another initial and informal study to examine the proportion of PDBNs that would cause the inference algorithms to crash. I then proceeded to modify my initial selection of parameters with the objective of generating a larger yield of usable PDBNs and repeated tests on the newly generated set of PDBNs. I repeated this evolutionary process until I was satisfied with the yield of usable PDBNs in a sample set.

I ran the inference algorithms over a set number of time steps, each time step representing an increment in time. The number of time steps I ran the inference algorithms over was an important factor because, if the number was too small, there may not be enough information to generate very accurate inference results. Conversely, if the number of time steps each inference algorithm is run over was too large, the experimentation phase of this research would take too long. I found that the ideal number of time steps to run each inference algorithm over was approximately 60 time steps.

The inference algorithms I examined in this research study can be separated into two groups: exact inference algorithms and approximate inference algorithms. The exact inference algorithms serve to provide an understanding of the use of inference algorithms in problem domains where exact inference is feasible and accuracy is much more important than speed of inference. The approximate inference algorithms serve to provide an understanding of the use of inference algorithms in problem domains where exact inference is infeasible or where both speed and accuracy are of great concern.

The inference algorithms I used in this study were taken from the Quiddity commercial software package created by IET, Inc (IET, 2003). The Quiddity software package provided the symbolic probabilistic inference algorithm and the Boyen-Koller inference algorithm utilizing the symbolic probabilistic inference algorithm. This software package also provided a particle filter inference algorithm that I used in this study.

Also, most of the inference algorithms tested in this research study have implementation variations that may affect their effectiveness. The PDBN inference is

being performed on may not be independent of the performance of these variants. The speed at which an inference algorithm runs and the accuracy of the results it produces can be controlled using attributes beyond those attributes defining the PDBN architecture. For this reason, different versions of each inference algorithm that have parameters beyond the PDBN to be used were tested at different levels of emphasis of speed at the expense of accuracy. I controlled these parameters at a series of levels such that each level should cause the algorithms to perform faster than the last. This increase in speed, however, was always at the expense of accuracy.

The Boyen-Koller and particle filtering inference algorithms have adjustable parameters that can affect inference performance and accuracy beyond the architecture of the PDBN that inference is being performed on. The Boyen-Koller inference algorithm requires one to define a set of clusters and, in effect, choose which arcs to remove when performing inference. Although there are heuristics and methods to aid in the choice of clusters, there are few standardized algorithms that work well in most cases. However, the choice of cluster placement and the size of the clusters are of tremendous importance in both the performance and accuracy of the inference algorithm. The particle filtering inference algorithm requires one to define the number of particles to be used in the Monte Carlo estimation of the PDBN state. The more particles used, the more accurate the inference algorithm is. However, this is at the cost of the speed at which inference can be done. Fewer particles increases speed at the expense of accuracy.

It is not the primary purpose of the study to determine the exact nature of the relationship between these other inference algorithm parameters and inference algorithm

effectiveness. However, these parameters may not be entirely independent of the factors that define a PDBN’s architecture, and hence, must be examined to some extent in this research study. Therefore, I varied a set of levels varying these parameters in the experiment. Both of these parameters allow the user of the inference algorithm to balance speed for accuracy. These levels are different gradations of speed emphasis at the expense of accuracy and are referred to throughout the experiment as “speed emphasis levels.”

Table 4 lists the 3 gradations of the speed emphasis levels categories for the Boyen-Koller inference algorithm and Table 5 lists the 3 gradations of the speed emphasis levels categories for the particle filtering inference algorithm. Each of the gradations listed in Table 5 is a range of values that would be selected at random when running the experiment. I used each of these gradations in the experiment when using the Boyen-Koller inference algorithm or the particle filtering inference algorithm respectively. The were chosen based on my experience with using each of these algorithms along with the expert judgment of Dr. Kathryn Laskey, advisor to this study.

Table 4: Speed Emphasis Levels for the Boyen-Koller Inference Algorithm

Average Cluster Size	Description
n / 5	The average cluster holds 20% of the nodes.
n / 8	The average cluster holds 12.5% of the nodes.
Fully Factored	All clusters contain only one node.

Table 5: Speed Emphasis Levels for the Particle Filtering Inference Algorithm

Number of Particles
5,000 – 10,000
1,000 – 5,000
0 – 1,000

The number of particles used in the test runs of this experiment to perform inference is rather small compared to the number of particles customarily used. A more reasonable number of particles to use with the particle filter inference algorithm would be at the order of magnitude of 10,000 or even 100,000. I used fewer particles than what is normally used in order to complete the experiments of this research study in a reasonable amount of time.

The process by which clusters were chosen for use with the Boyen-Koller algorithm is straightforward for the speed emphasis level that generates fully factored clusters. When a cluster can contain multiple nodes, however, the choice of nodes falling in a particular cluster is less straightforward and may have a great deal of influence over the speed and accuracy of the Boyen-Koller inference algorithm. In this research study, node selection per cluster was chosen using a trial and error algorithm. For a given speed emphasis level, all clusters were of equal size where possible and varied by no more than one node at the most. A set of cluster combinations were produced randomly where the number of cluster combinations was ten times the number of nodes in the interface. The nodes in the interface chosen to be in a particular cluster were chosen at random without replacement. The chance of any node being in any cluster is equal. Choosing nodes for a cluster with replacement may lead to clusters overlapping. This is a perfectly valid

option. However, overlapping clusters risk violating the running intersection property. Because of time and resource constraints, I decided to not allow overlapping clusters and opted for a much simpler cluster selection algorithm. Each cluster combination was then evaluated by examining the sum of the correlations between each node not in the same cluster. The correlation between two nodes was calculated regardless of whether there exists an arc between those two nodes. The cluster combination with the lowest sum of correlations was the chosen cluster combination.

I choose this cluster combination because it is the one I believe to have the least difficulty of all the cluster sets of handling the independence assumption between nodes of different clusters that the Boyen-Koller algorithm makes. When the Boyen-Koller inference algorithm performs inference, it performs inference calculations as if the arcs between nodes of different clusters do not exist. In this way, the inference calculations are often simplified. The choice of clusters, however, plays an important role in how well the Boyen-Koller algorithm is able to do this. Ideally, the correlation between nodes in different clusters should be as low as possible and arcs between nodes of different clusters should represent as weak a relationship as possible. The cluster set that has the least total correlation between nodes not in the same cluster is chosen because it should meet these criteria reasonably well.

It is important to note that this method of cluster selection puts the Boyen-Koller inference algorithm at a disadvantage regarding its effectiveness. This method does not allow a great deal of variety in cluster size or arrangement, and hence, does not represent all possible arrangements of cluster organization that a modeler may utilize.

Furthermore, a knowledge engineer developing a Bayesian network to simulate a real world system will use his or her understanding of the problem domain to choose a cluster set that best suits the problem he or she is modeling. This method of random cluster creation does not reflect this fact and the cluster arrangement has a great deal of influence over the effectiveness of the Boyen-Koller algorithm. There is a distinct possibility that this method will choose a cluster arrangement that will cause the inference algorithm to perform worse than a cluster arrangement that would be chosen by a domain expert or a knowledge engineer. This particular method of cluster selection was chosen for speed of use and easy implementation. A more complex implementation of cluster selection would be too difficult to develop and use given the resources available for this research project and is presently beyond the defined scope of this research. In fact, cluster choice in the Boyen-Koller inference algorithm is itself a problem worthy of study. I believe that this trial and error algorithm still mitigated this loss of speed and accuracy somewhat so that the bias against the Boyen-Koller algorithm was not significant.

For the purposes of simplicity, I ran all the test runs of the experiment on the same computer. However, for this research to be of any use, one must be able to generalize these results to any computer. Any set of computers capable of performing inference on a PDBN can vary greatly in the resources they provide, and hence, can potentially provide different performance results (although accuracy results will naturally remain the same regardless of the computer used).

Although I cannot show that the relationship between the performance results on any two computers is linear, I will show that the performance results on the different

computers I used are linearly related. To verify this hypothesis, I executed an initial study consisting of a series of test runs on several different computers. In this study, I generated a sample set of PDBNs to go through a test run on each computer. I then compared the speed results derived from each PDBN on each computer to each other.

The initial study I ran consisted of four computers of various speeds, power, and memory amounts. Table 6 lists various speed and power statistics for each of the computers used in this study. The computers are listed in order of descending power. The first column labels the computer. The second column describes the speed and type of the CPU in the computer. The last column gives the amount of RAM in the computer.

Figure 6 gives the results of the study. The graph in Figure 6 compares the speed at which the HP, Dell, and E-Tower computers performed in comparison to the Laptop computer. I examined each PDBN individually and I compared the speed results for each PDBN on the HP, Dell, and E-Tower computers to the speed results from the Laptop computer. The graph is a scatter plot where each point in the graph represents the comparison between the Laptop computer and one of the other three computers. The x-axis represents the average time per time step for the Laptop and the y-axis represents the average time per time step for each of the other 3 computers.

Table 6: Speed and Power Statistics on Four Computers used in Study

	CPU	RAM
Laptop	1.33 GHz AMD Athlon	240 MB
HP	1.2 GHz AMD Athlon	256 MB
Dell	798 MGz Intel Pentium III	256 MB
E-Tower	400 MHz Intel Celeron	160 MB

Comparison of Results on Various Computers

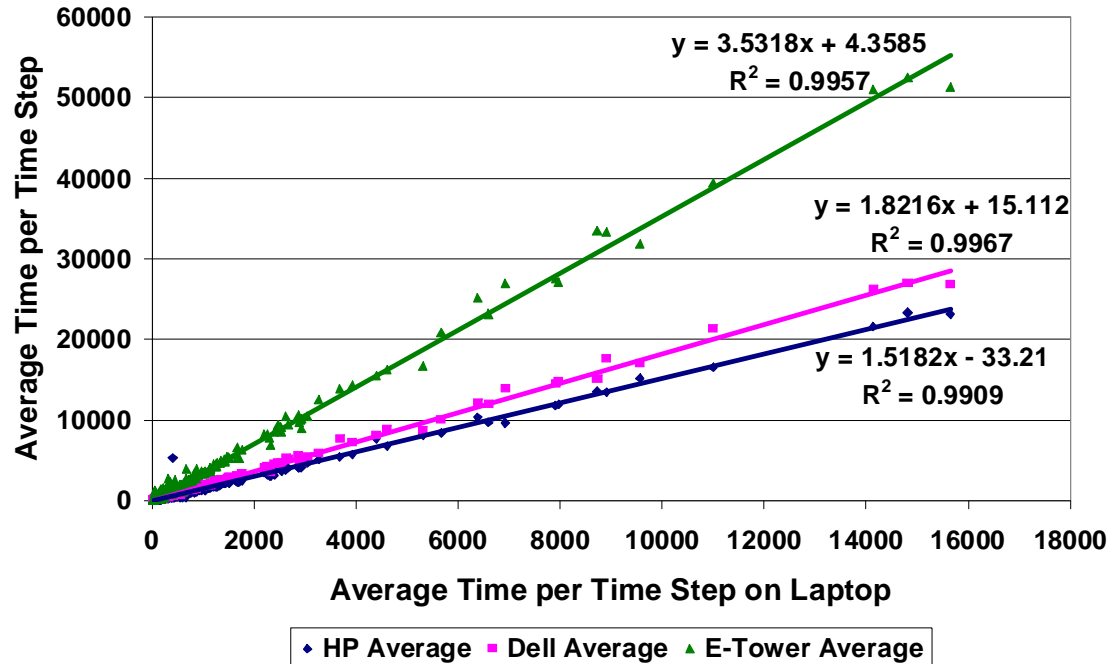


Figure 6: Comparison of Results on Various Computers

The plots for each of the comparisons between the Laptop and the other three computers are all linear in the graph. Each point for each computer compared to the Laptop computer lies close to the fitted line (produced by Excel). Furthermore, the R^2 values (also produced by Excel) are all above 0.99. It is, therefore, reasonable to assume that the performance results on the different computers I used are linearly related. The performance of the Laptop in comparison to each of the three other computers is certainly linear and, if each computer is linearly related to the Laptop computer, it is reasonable to expect them to be linearly related to each other as well.

2.2 Experimental Methodology

The random set of generated PDBNs used in this research study is meant to provide breadth to the results of this research because this sample set contains a large number and highly varied sample PDBNs. This methodology is in contrast to one where I would get a smaller sample set of representative PDBNs within a particular problem domain. By randomly generating my sample set of PDBNs, I am more likely to cover the full spectrum of possible PDBNs and less likely to produce results that are biased by a problem domain I drew samples from.

I drew the sample set of PDBNs from randomly generated networks from a random PDBN generator. The set of randomly generated networks consisted of 250 PDBNs. I developed the random PDBN generator before experimentation began and generated random PDBNs using the experimental factors listed in Table 2. For each experimental factor in listed in Table 2, I used the distribution defined in Table 3 when using the

random PDBN generator. This random PDBN generator allows the user to control the distributions of the experimental factors that define the attributes of each PDBN in the generated sample set. The random PDBN generator I developed can also measure empirically the values of each experimental factor, as well as other factors, for each PDBN in the sample set.

Once a sample PDBN was selected from the random PDBN generator, it was categorized based on the factors described in Tables 1 and 2. This categorization was necessary for experimental purposes to determine what factors most influence a particular inference algorithm. Once there was an adequately sized sample set of PDBNs and these Bayesian networks were categorized by the factors of the experiment defined in Table 2, the test runs of the experiment began.

Once a PDBN was selected and properly categorized, the formal test run began. Each test run began with the generation of a base truth set of cases. This is essentially a set of values for each variable over each time step to represent the actual value of that variable at that time step. This base truth set of cases varies for each PDBN but not over each inference algorithm that utilizes the same PDBN. This was used later in data collection and analysis to help determine the accuracy of the inference algorithm for this Bayesian network. Once the base truth set of cases was generated, the inference algorithms could be used to perform inference on the sample PDBN over 60 time steps. Data examining the performance and accuracy of the inference algorithm were collected while the inference algorithm processes the PDBN. The data collected is described in the next section and listed in Table 7.

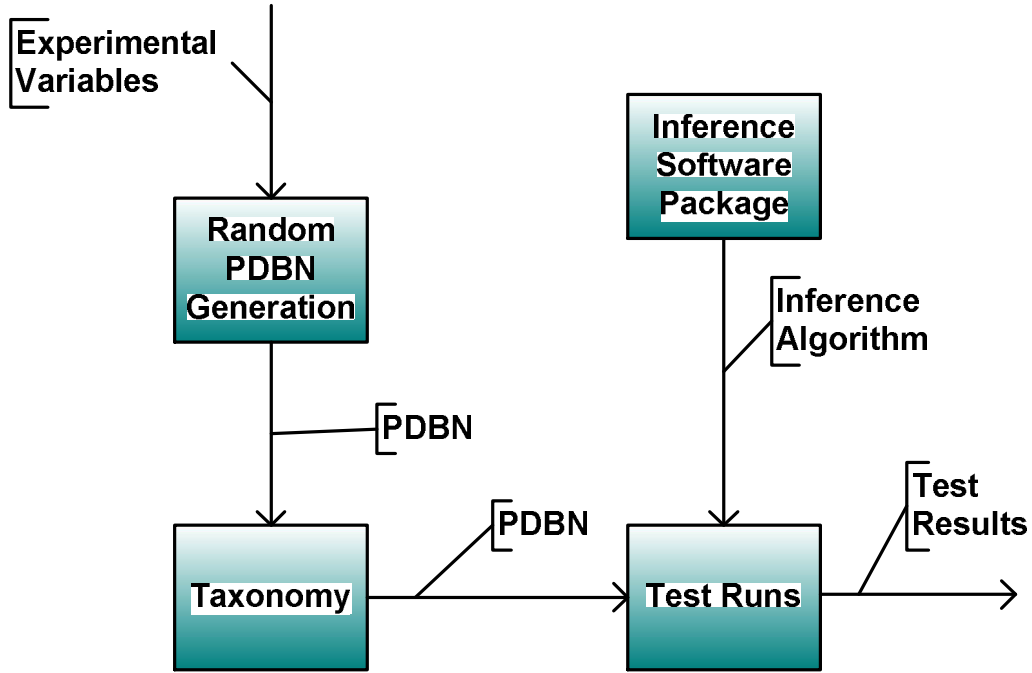


Figure 7: Formal Experimentation Flow Chart

Within the base truth case set generation stage, a set of cases for all nodes in the PDBN being tested were generated for each time step through Monte Carlo simulation. These cases represent the actual value of each variable over each time step. Then the inference algorithm was run on the PDBN. For each time step, the set of cases in that time step were applied to the nodes of that time step that are chosen to have evidence. According to the constraints of this research study, only dynamic non-transitional nodes can have evidence. The set of nodes to have evidence were randomly decided ahead of time and within each time step there was a 1% probability that a particular node will not get any evidence for that time step only. Each node had an equal probability of being an evidence node. The max a posteriori value was calculated for each node without

evidence using the evidence given at that time step. The max a posteriori value was collected to represent the estimated value of that variable and the set of cases collected before the inference algorithm is applied was used to represent actual value of each variable in the Bayesian network.

During the formal experiment, it was also necessary to designate a class variable (and hence, a class node) among the nodes in the partially dynamic Bayesian network. A class variable is the single variable in a Bayesian network modeling a class discrimination problem that represents the class of the object being examined. The class node is the node representing this variable. This node is usually static and discrete. Because the networks generated for this research study do not model a specific real world problem, there is no real class node in the PDBNs in the sample sets. Therefore, a static root node was chosen to simulate the class variable for each PDBN in the sample set.

I then used this class variable as the point of comparison for the accuracy metrics. With each of the approximate inference algorithms, I compared the distribution of the class variable given by the approximate inference algorithm to the exact distribution of this same variable. The exact distribution of the class variable was derived using the SPI algorithm, which is an exact inference algorithm.

The designation of a class variable was useful for the purpose of determining the number of time steps necessary to perform an accurate object classification and for measuring accuracy metrics dealing specifically with the class variable. The former objective was necessary in determining the number of time steps a test run should occur. In any object classification problem, the class variable will start off with several likely

candidates, each with a different probability. After several time steps where evidence is applied and more is understood about the object being identified, certain possible classifications of the object become more likely than the others. However, it may not be possible even after many time steps to determine the correct classification with certainty. There may be a period of transient behavior of the class node, in which several classifications may vie for the position of the most likely classification for the object being examined. For some situations, the class node may eventually reach a point of near-certainty for a given class. In other situations the class node may reach a limiting probability between zero and one, oscillate among possible classifications, or enter some other pattern that will continue indefinitely.

The number of time steps to conduct inference over a test run should ideally allow just enough time for the class variable to reach a stable and unchanging pattern in its state transition for all test runs in the experiment. If the class variable is not given enough time to reach a state of equilibrium, the accuracy metric reported for that test run will be adversely affected. Through informal tests, I was able to determine that 60 time steps provided ample time for any test run of a PDBN within the scope of this research study under any of the inference algorithms used in this research study to have the class variable of the PDBN to reach a state of equilibrium.

3. Experimental Results

3.1 General Analysis Methodology

A suitable inference algorithm is one that is fast enough to run in real time and accurate enough to provide meaningful and trustworthy results given a reasonable resource allocation. Therefore, the metric categories by which each inference algorithm was judged were speed and accuracy. Table 7 lists all the assessment metrics for each of these value objectives as well as a brief description of each. Most of the assessment metrics come from Section 4 of the “Test Plan for Evaluating Approximate Inference Algorithms” (IET, 2002).

Table 7: Assessment Metrics (IET, 2002)

	Name	Description
Speed Metrics		
	Average time per time step	How much time on average was required to perform inference over each time step.
	Variance in time per time step	Variance in time to perform inference over each time step.
Accuracy Metrics		
	Root mean squared error	The square root of the sum of the squares of the per-value differences between the exact and approximate distribution values $\sqrt{\left(\sum_i q_i - p_i\right)^2}$

I examined the speed of the inference algorithm per time step per test run. For each time step, I measured the amount of time to complete the time step for each inference algorithm.

I also examined the accuracy of the inference algorithm per time step and per test run. I computed accuracy values for all nodes not designated to be evidence nodes. For evidence nodes that did not collect evidence for a time step, however, I did not examine these nodes for accuracy. For each test run, I used the average root mean squared error over each time step for each inference algorithm to determine how much the approximate distribution varies from the exact distribution.

The data that I collected each test run to calculate what each of these assessment measures was:

- The time required to perform the time step inference.
- The posterior distribution on each node at each time step.
- The distribution of each node after the final time step.
- Base truth value of each node over each time step.

I used the posterior distributions derived from the exact inference algorithms as the basis of comparison for the approximate inference algorithms to analyze the accuracy of these inference algorithms. While even an exact inference algorithm can give false positives and false negatives the posterior probability distributions it provides will be accurate and a valid basis of comparison for approximate inference algorithms. In the cases where the particular class of PDBN was intractable for the exact inference algorithms being used in this study but not the approximate inference algorithms, it was

impossible to calculate all the accuracy assessment metrics and was treated as missing data. Thus, I only used PDBNs that were tractable for both the SPI and approximate inference algorithms when taking accuracy measures.

I developed my models using linear regression. I used the linear regression package that was available with Microsoft Excel as well as the statistical package R to conduct linear regression. I used both Microsoft Excel and R to perform linear regression. I also used R to build correlation matrices, calculate variation inflation factors, and calculate Cook's distance statistics to better understand and support my statistical models.

I used linear regression to develop the statistical models dealing with the accuracy of inference, the average speed of inference per time step, and the average variance of the average speed of inference per time step. The regression equation I used is shown in Equation 3.

$$Y = \alpha + \sum_i \beta_i x_i + \varepsilon \quad \text{Equation 3}$$

Here Y is the dependent variable, x_i is the i th independent variable, β_i is the coefficient of the i th independent variable, α is the y intercept, and ε is the random error term.

However, there is reason to expect that the relationship between the dependent variable and the average speed of inference per time step may be nonlinear. Because inference on Bayesian networks is an NP-Hard problem (Cooper 1990), it would stand to reason that a linear increase in complexity would result in an exponential increase in time of inference. Therefore, I also considered an exponential regression model. To develop an exponential regression model, I performed linear regression using the alternate regression equation shown in Equation 4.

$$\ln(Y) = \alpha + \sum_i \beta_i x_i + \varepsilon \quad \text{Equation 4}$$

Thus, I performed linear regression using the natural logarithm of the dependent variable. The case when the dependent variable isn't transformed for this regression equation is shown in Equation 5.

$$Y = e^\alpha \prod_i (e^{\beta_i})^{x_i} e^\varepsilon \quad \text{Equation 5}$$

The situation where an exponential regression model would likely be the better model would be in the case of the Boyen-Koller inference algorithm when one uses it in conjunction with the SPI exact inference algorithm. One possible factor that one could use to measure the complexity of a Bayesian network is the number of nodes in that network. Adding a single node to the Bayesian network will have a multiplicative effect on the size of the CPTs of every node that is a child of the node that was added. The speed at which the SPI inference algorithm is able to do inference is likely linked to the size of the CPT tables of the nodes in the Bayesian network. Therefore, it stands to reason that adding a node to a Bayesian network will result in an exponential increase in the time of inference for that Bayesian network when using the SPI algorithm.

It is unclear whether an exponential model would be appropriate for modeling the speed of inference of a particle filter inference algorithm. Unlike most inference algorithms, the particle filter inference algorithm may not grow exponentially in time to perform inference as the complexity of the PDBN grows linearly (Kuo 2005). However, I believe it is still worthwhile to investigate the possibility that the speed of inference for the particle filter inference algorithm could be modeled with an exponential model.

For this reason, I also developed exponential statistical models dealing with the average speed of inference per time step. I did this by first linearizing the exponential statistical model by using the natural logarithm of my dependent variable. Then I performed a linear regression in which the dependent variable was the natural logarithm of the average speed of inference per time step.

My initial attempts at regression analysis for each performance and accuracy metric for each inference algorithm utilized all the independent variables in Table 1 regardless of their theoretical relevance to the model I was developing. I removed factors as independent variables if they correlated beyond a threshold of 0.15 to other independent variables that fit better in the model. Even after this step to reduce collinearity among the independent variables, I was left with a relatively large number of independent variables with which to perform regression over. In all situations, the resulting models were poor. In many scenarios, there were significant factors in the model with negative coefficients where one would expect a positive coefficient. In addition, minor changes to the independent variables used often lead to dramatic changes in which factors were significant. Furthermore, I consistently had one or more independent variables with variance inflation factors above 10 in each of the models (Kleinbaum, 1988).

My conclusion was that there is multicollinearity between some of my independent variables and that the method by which I selected the independent variables to use in a regression analysis was flawed. To reduce the multicollinearity between my independent variables, I used only a subset of the factors in Table 1, selecting only those factors that I could justify as having any likelihood of affecting the dependent variable I was

examining. Along with this approach, I also examined the correlation matrix of every factor I used in the regression analysis as well as each factor's variance inflation factor. In this way, I could minimize the multicollinearity between my independent variables.

I validated my models over a sample set of 250 PDBNs. This amounted to a maximum of 1500 data points when examining the performance of each of the particle filter inference algorithms and 2000 data points when examining the performance of the Boyen-Koller inference algorithms. In practice, however, I used fewer than the maximum potential number of data points for evaluating an individual inference algorithm. Fewer than the maximum number of data points were available for evaluating each inference algorithm because inference over some of the sample PDBNs was not tractable or timely for some of the scenarios the PDBNs were tested under.

When performing linear regression to determine the factors to use in my models, I used a reduced training set of half the available data points. I further reduced my dataset by removing repeat observations¹, or pairs of observations where each of the independent variables and the dependent variable were identical. However, this reduction was always small. I then tested the models over the entire available dataset. By performing regression analysis over half of the data points, I believe I had enough data during the regression step to prevent over fitting. At the same time, I believe I had enough data left over to test the model produced by the regression test accurately.

¹ Here a repeat observation is an observation that is likely to be generated under the same scenario and PDBN as another observation. I determined if an observation was likely to be a repeat observation if that observation shared the same values of its dependent variables and independent variable as another factor.

3.2 Analysis and Results for the Particle Filter Inference Algorithm Speed of Inference

For the speed of inference models and the variance of speed of inference models for the particle filter inference algorithm I used the factors:

- Average States per Node
- Average CPT Size
- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Number of Static Nodes
- Number of Particles

The number of particles used by the particle filter is an obvious choice as a factor that influences the speed of inference of this inference algorithm. The particle filter inference algorithm is a Monte Carlo method, and thus, the algorithm must spend time creating or adding to each particle every time step.

How long the algorithm takes to create or add to a particle depends on the number of nodes in the Bayesian network because each node must randomly selected its value for each particle. Therefore, it also made sense to include the number of each class of nodes in the PDBN as an independent variable used in the regression analysis. The factors for each of the three classes of nodes are: the Number of Dynamic Transitional Nodes factor, the Number of Dynamic Non-Transitional Nodes factor, and the Number of Static Nodes factor.

When randomly selecting the value of a node, the particle filter inference algorithm must choose a value among all the possible states of that node using the conditional

probability distribution of that node. Although the time to randomly select the value of a node is not necessarily related to the number of states in that node or the size of that node's CPT table, these factors were included incase there still did exist a significant relationship.

Furthermore, the CPT tables of all the nodes contribute the greatest to the amount of memory a PDBN takes up. The more memory resources a PDBN takes up, the fewer resources are available for the inference algorithm. This could conceivably have a negative effect on inference algorithm performance.

When performing regression with the exponential model, I combined the three factors dealing with the number of each class of nodes in a PDBN. Had I kept these factors separate when performing regression with the exponential model, these factors would have been multiplied together rather than summed. However, it is likely that the sum of these factors, and not the product of these factors that affect the speed of inference for the particle filter inference algorithm. The particle filter inference algorithm randomly selects a value for each node for each particle and only once for each node for each particle. Thus, it makes more sense to combine these thee factors into the Number of Nodes factor and have that factor multiplied with the Number of Particles factor when developing the exponential model.

The first inference algorithm I examined was an exponential model of the speed of inference for the particle filter inference algorithm. The first set of factors I performed regression with was:

- Average States per Node

- Average CPT Size
- Number of Nodes
- Number of Particles

This model had a clear pattern in the residuals graph for the Number of Particles factor. This graph can be seen in Figure 12 in Appendix A. The pattern appears to suggest a logarithmic relationship between the number of particles used by the inference algorithm and the natural logarithm of the speed of inference for the particle filter inference algorithm.

The Average States per Node factor was also not significant in this model. I performed regression again over the same set of factors excluding the Average States per Node factor. Despite this change, there was still a clear pattern in the residuals graph for the Number of Particles factor. The graph was nearly identical to the one seen in Figure 12.

Based on the graph in Figure 12, I adjusted the previous model by using the natural logarithm of the Number of Particles factor as an independent variable. Thus, the set of factors I performed regression over became:

- Average States per Node
- Average CPT Size
- Number of Nodes
- Natural Logarithm of the Number of Particles

In this model, all factors including the Average States per Node factor were significant. The graph of the residuals versus the natural logarithm of the Number of Particles factor

in this model does not appear to show a pattern as the Number of Particles factor did in the previous model. This graph is shown in Figure 13 in Appendix A. However, there appeared to be a pattern now in the residuals graph for the Average CPT Size as seen in Figure 15 in Appendix A. As before, the pattern appeared to be logarithmic in nature.

In this model, several outliers had to be removed because they were beyond the threshold of 0.5 for Cook's distance statistic (Lorenz, 1987). Thus, these data points disproportionately influenced the regression and needed to be removed so the regression could be more accurate. One can see these outliers best in Figure 14, which is the graph of the residuals versus the natural logarithm of the Number of Particles factor before these outliers were removed. The outlying data points in are circled. From this figure, one can see that these outliers lie outside the general trend of the plot. Only six data points were removed and after these data points were removed, no further outliers were shown as being overly influential according to Cook's distance statistic.

In the next model, I used the factors:

- Average States per Node
- Natural Logarithm of the Average CPT Size
- Number of Nodes
- Number of Particles

I used the natural logarithm of the Average CPT Size factor in response to the pattern that appears in Figure 15. As in the first exponential model I developed to predict the speed of inference for the particle filter inference algorithm, there was an apparent pattern in the residuals graph for the Number of Particles factor. This graph is seen in Figure 16 in

Appendix A. As before, the pattern suggested a logarithmic relationship between the number of particles used by the inference algorithm and the natural logarithm of the speed of inference for the particle filter inference algorithm

Also similar to the first exponential model I developed, the Average States per Node factor was also not significant in this model. I performed regression again over the same set of factors excluding the Average States per Node factor. As before, there was still a clear pattern in the residuals graph for the Number of Particles factor. The graph was nearly identical to the one seen in Figure 16.

The next model I developed for predicting the speed of inference for the particle filter inference algorithm used the factors:

- Average States per Node
- Natural Logarithm of the Average CPT Size
- Number of Nodes
- Natural Logarithm of the Number of Particles

None of the residual graphs for any of the factors listed above displayed any apparent patterns as seen in Figure 17, Figure 18, Figure 19, and Figure 20 in Appendix A.

In this model, several outliers had to be removed because they were beyond the threshold of 0.5 for Cook's distance statistic. Thus, these data points disproportionately influenced the regression and needed to be removed so the regression could be more accurate. One can see these outliers in Figure 21, which is the graph of the residuals versus the natural logarithm of the Number of Particles factor before these outliers were removed. The outlying data points in are circled. From this figure, one can see that these

outliers lie outside the general trend of the plot. Only six data points were removed and after these data points were removed, no further outliers were shown as being overly influential according to Cook's distance statistic.

Figure 22 in Appendix A shows the model's fit to the actual time results while Figure 23 in Appendix A shows this same graph in logarithmic scale. Figure 24 in Appendix A shows the residual plot for this exponential model. From the graphs in Figure 22 through Figure 24, this exponential model appears to be a good predictor of the actual speed of inference for the particle filter inference algorithm. Although there is a handful of outliers in the graph, the results of Cook's distance statistic suggests that none of these outliers were overly influential on the results of the regression analysis. Also, it appears that as the average time to perform inference grows linearly, the error of this model grows slightly. This can be seen from Figure 22. However, this linear growth in error is only slight.

A Normal Q-Q plot of the residuals, as seen in Figure 25 in Appendix A, supports the assumption that this model's error is normal. There are outliers at the right tail of the normal distribution but, for the main body of the data, the model's error appears to match the normal distribution closely. The Q-Q plot along with the residual and model fit graphs above in Figure 17 through Figure 25 all support the Gauss-Markov assumptions. The model's error is likely to be normal with a mean near zero as seen from the Q-Q plot. Furthermore, the random errors of the model are all likely to be uncorrelated as no factor has a variance inflation factor above 2. Finally, the lack of any noticeable patterns in any

of the residual graphs for each of the independent variables and the dependent variable suggests that random errors of the model are homoscedastic.

For the linear model of speed of inference for the particle filter inference algorithm, the factors I used were:

- Average States Per Node
- Average CPT Size
- Number of Static Nodes
- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Number of Particles

This model proved to be a very good model as none of the residual graphs for any of the factors listed above displayed any apparent patterns as seen in Figure 26 through Figure 31 in Appendix A.

A pattern was found for the dependent variable. Figure 32 in Appendix A shows the model's fit to the actual time results. Figure 33 in Appendix A shows the residual plot for this exponential model. Although the model's fit to the actual time results appears to be an ideal fit, the residual plot in Figure 33 appears to suggest a logarithmic or quadratic relationship.

Nonetheless, the graph in Figure 32 supports the belief that this linear model is a better predictor of the actual speed of inference for the particle filter inference algorithm than the other models explored in this research study for this dependent variable. Figure 33 suggests that some heteroscedasticity may exist but it still may not be substantial

enough to disrupt the regression analysis used to generate this model. Although there are a handful of outliers in the graph, the results of Cook's distance statistic suggests that none of these outliers was overly influential on the results of the regression analysis.

A Normal Q-Q plot of the residuals, as seen in Figure 34 in Appendix A, supports the assumption that this model's error is normal. Furthermore, this Q-Q plot matches the normal distribution better than the Q-Q plot in Figure 25 for the exponential model above. As in the exponential model, the Q-Q plot along with the residual and model fit graphs above in Figure 26 through Figure 34 all support the Gauss-Markov assumptions. The model's error is likely to be normal with a mean near zero as seen from the Q-Q plot. Furthermore, the random errors of the model are all likely to be uncorrelated as no factor has a variance inflation factor above 2. Finally, the lack of any noticeable patterns in any of the residual graphs for each of the independent variables and the dependent variable suggests that random errors of the model are homoscedastic.

Another concern with regard to this linear model is that there was a relatively high correlation between the Number of Dynamic Transitional Nodes and Average CPT Size factors. These two factors had a correlation of 0.33 where my correlation threshold is 0.3. I endeavored to avoid any pair of factors being correlated beyond a certain threshold because a pair of factors that are too highly correlated may result in multicollinearity problems in the regression results. A correlation threshold of 0.3 should prevent any two factors from being collinear. Nonetheless, the value of 0.3 was chosen arbitrarily so a correlation of 0.33 may still not be large enough to disrupt the regression analysis for this

model. This is supported by the fact that variance inflation for all the factors of this model were well below 2.

To address the above concern, I generated a linear model that used the total number of nodes as a factor instead of the number of each class of nodes. Thus, the factors I used in this model were:

- Average States Per Node
- Average CPT Size
- Number of Nodes
- Number of Particles

The residual graphs, model fit plots, and Q-Q plot for this model were nearly identical to that of the first linear model. The residual graph for the Number of Nodes factor also had no apparent structure as seen in Figure 35 in Appendix A. The R square statistic for the test set was lower than the previous linear model, however, at 0.974613.. Thus, it is probably better to keep the three factors dealing with node type separate for the linear model of the speed of inference of the particle filter inference algorithm.

3.3 Analysis and Results for the Particle Filter Inference Algorithm Variance of Speed of Inference

I used the same set of factors when developing both the speed of inference and variance of the speed of inference for the particle filter inference algorithm. These factors were:

- Average States per Node

- Average CPT Size
- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Number of Static Nodes
- Number of Particles

I decided to use the same set of factors because of the strong relationship between the speed of inference and the variance of the speed of inference. The relationship between these two factors becomes apparent in the graph in Figure 8. Thus, as the time to perform inference with the particle filter inference algorithm increases, so does the variance of the time to perform inference. I believe that this relationship entails that the physics behind the speed of inference and the variance of the speed of inference for the particle filter algorithm are similar. Thus, I believe the models that predict these two metrics would likely be equally similar as well as the factors involved.

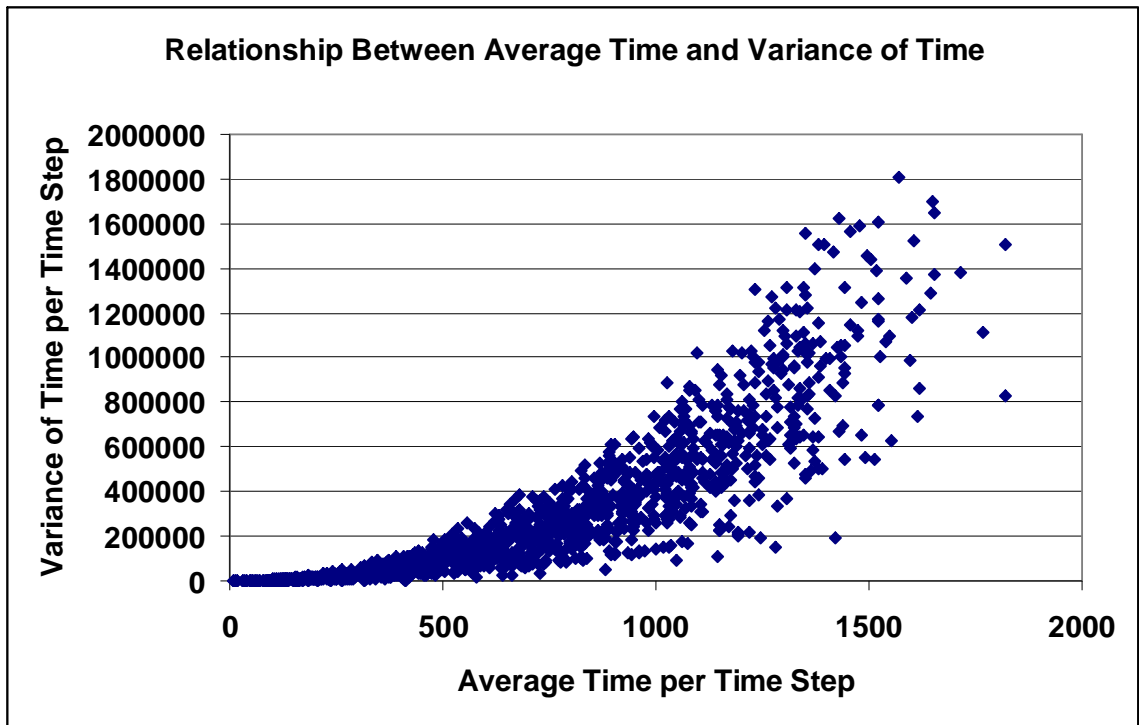


Figure 8: Relationship Between Average Time per Time Step and Variance of Time per Time Step for the Particle Filter Inference Algorithm

The first inference algorithm I examined was an exponential model of the variance of the speed of inference for the particle filter inference algorithm. The first set of factors I performed regression with was:

- Average States per Node
- Average CPT Size
- Number of Nodes
- Number of Particles

This model had a clear pattern in the residuals graph for the Number of Particles factor.

This graph can be seen in Figure 36 in Appendix B. The pattern appears to suggest a logarithmic relationship between the number of particles used by the inference algorithm

and the natural logarithm of the variance of the speed of inference for the particle filter inference algorithm.

The Average States per Node factor was also not significant in this model. I performed regression again over the same set of factors excluding the Average States per Node factor. Despite this change, there was still a clear pattern in the residuals graph for the Number of Particles factor. The graph was nearly identical to the one seen in Figure 36.

Based on the graph in Figure 12, I adjusted the previous model by using the natural logarithm of the Number of Particles factor as an independent variable. Thus, the set of factors I performed regression over became:

- Average States per Node
- Average CPT Size
- Number of Nodes
- Natural Logarithm of the Number of Particles

In this model, all factors including the Average States per Node factor were significant. The graph of the residuals versus the natural logarithm of the Number of Particles factor in this model does not appear to show a pattern as the Number of Particles factor did in the previous model. This graph is shown in Figure 37 in Appendix B. However, there appeared to be a pattern now in the residuals graph for the Average CPT Size as seen in Figure 38 in Appendix B. As before, the pattern appeared to be logarithmic in nature.

In this model, several outliers had to be removed because they were beyond the threshold of 0.5 for Cook's distance statistic. Thus, these data points disproportionately

influenced the regression and needed to be removed so the regression could be more accurate. After these data points were removed, no further outliers were shown as being overly influential according to Cook's distance statistic.

The next model I developed for predicting the variance of the speed of inference for the particle filter inference algorithm used the factors:

- Average States per Node
- Natural Logarithm of the Average CPT Size
- Number of Nodes
- Natural Logarithm of the Number of Particles

None of the residual graphs for any of the factors listed above displayed any apparent patterns as seen in Figure 39, Figure 40, Figure 41, and Figure 42 in Appendix B.

In this model, several outliers had to be removed because they were beyond the threshold of 0.5 for Cook's distance statistic. Thus, these data points disproportionately influenced the regression and needed to be removed so the regression could be more accurate. After these data points were removed, no further outliers were shown as being overly influential according to Cook's distance statistic.

Figure 43 in Appendix B shows the model's fit to the actual variance of time results while Figure 44 in Appendix B shows this same graph in logarithmic scale. Figure 45 in Appendix B shows the residual plot for this exponential model. From the graphs in Figure 43 through Figure 45, this exponential model appears to be a decent predictor of the actual variance of the speed of inference for the particle filter inference algorithm. Although there is a number of outliers in the graph of which several are quite distant, the

results of Cook's distance statistic suggests that none of these outliers were overly influential on the results of the regression analysis. Also, it appears that as the variance of the average time to perform inference grows linearly, the error of this model grows as well. This can be seen best in Figure 43.

The Normal Q-Q plot of the residuals, as seen in Figure 46 in Appendix B, does not support the assumption that this model's error is normal. The distribution is very heavy on the right tail. The distribution is also heavy on the left tail, although to a lesser degree. The Q-Q plot along with the residual and model fit graphs above in Figure 43 through Figure 45 do not support all the Gauss-Markov assumptions. Although the model is likely to have a mean error near zero, the model's error distribution is not likely to be normal. However, the random errors of the model are all likely to be uncorrelated as no factor has a variance inflation factor above 2. Also, the lack of any noticeable patterns in any of the residual graphs for each of the independent variables suggests that random errors of the independent variables are homoscedastic. The residuals graph for the dependent variable appears to have a pattern in it such that the model appears to underestimate the dependent variable for higher values of the model.

There also appears to be a separate cluster of data points among the residual graph shown in Figure 45 and the model's fit to the actual time results when plotted in logarithmic scale shown in Figure 44. This suggests that two distinct and separate phenomena may be represented in the dataset and that two separate statistical models are necessary to model these two phenomenon. However, due to time constraints, I was unable to explore this possibility further.

For the linear model of the variance of the speed of inference for the particle filter inference algorithm, the factors I used were:

- Average States Per Node
- Average CPT Size
- Number of Static Nodes
- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Number of Particles

This model had a pattern in the residuals graph for the Number of Particles factor. This graph can be seen in Figure 47 in Appendix B. The pattern appears to suggest either a logarithmic or a quadratic relationship between the number of particles used by the inference algorithm and the natural logarithm of the variance of the speed of inference for the particle filter inference algorithm.

The Average CPT Size factor was also not significant in this model. I performed regression again over the same set of factors excluding the Average States per Node factor. Despite this change, there was still a clear pattern in the residuals graph for the Number of Particles factor. The graph was nearly identical to the one seen in Figure 47.

Based on the graph in Figure 12, I adjusted the previous model by using the natural logarithm of the Number of Particles factor as an independent variable. Thus, the set of factors I performed regression over became:

- Average States Per Node
- Average CPT Size

- Number of Static Nodes
- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Natural Logarithm of the Number of Particles

This model was far worse than the last one, however, with a decrease in the mode's R Square value instead of an increase. More importantly, the pattern seen in the residuals graph for the Number of Particles factor in the last model become more drastic in this model. This graph can be seen in Figure 48 in Appendix B.

Based on the results seen in Figure 48, it appears the pattern seen in the residuals graph for the Number of Particles factor of the last model is more likely to be quadratic in nature. Thus, the set of factors used in my next model were:

- Average States Per Node
- Average CPT Size
- Number of Static Nodes
- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Number of Particles Squared

This model had no apparent pattern in any of the residual graphs for the independent variables as seen in Figure 49 through Figure 54 in Appendix B. However, the Average CPT Size factor was found to not be significant. Thus, a final linear model was developed using on these factors:

- Average States Per Node

- Number of Static Nodes
- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Number of Particles Squared

Figure 55 in Appendix B shows the model's fit to the actual variance of time. Figure 56 in Appendix B shows the residual plot for this exponential model. From the graphs in Figure 55 and Figure 56, this exponential model appears to be a poor predictor of the actual variance of the speed of inference for the particle filter inference algorithm. An obvious pattern can be seen from these two graphs is likely to be exponential. No such pattern exists among the residual graphs of the independent variables in Figure 49 through Figure 54, however, suggesting that the heteroscedasticity lies in the dependent variable itself.

The Normal Q-Q plot of the residuals, as seen in Figure 57 in Appendix B, does not support the assumption that this model's error is normal. The distribution is heavy on the both tails of the distribution. The Q-Q plot along with the residual and model fit graphs above in Figure 55 and Figure 56 do not support all the Gauss-Markov assumptions. The model does not have a mean error near zero and the model's error distribution is not likely to be normal. Nonetheless, the random errors of the model are all likely to be uncorrelated as no factor has a variance inflation factor above 2. Also, the lack of any noticeable patterns in any of the residual graphs for each of the independent variables suggests that random errors of the independent variables are homoscedastic. The residuals graph for the dependent variable, however, appears to have a pattern in it such

that the model appears to underestimate the dependent variable for higher values of the model and the error grows exponentially as the model values grow linearly.

3.4 Analysis and Results for the Boyen-Koller Inference Algorithm Speed of Inference

For the speed of inference models and the variance of speed of inference models for the Boyen-Koller inference algorithm over SPI ²I used the factors:

- Average States per Node
- Average CPT Size
- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Number of Static Nodes
- Average Cluster Size

Just as the number of particles is a necessary parameter for the particle filter inference algorithm, the designation of nodes in each cluster is a necessary parameter for the Boyen-Koller inference algorithm. In these models, the Average Cluster Size factor is the metric I used to measure the effect these clusters have on the speed of inference on the Boyen-Koller inference algorithm over SPI. This metric is a proxy measure as the true indicator of the effect the clusters in the Boyen-Koller inference algorithm have on speed of inference is the change in the CPT sizes due to the removal of intercluster arcs. However, this metric is far more difficult to calculate and impossible to calculate in an

² Boyen-Koller inference algorithm when one uses it in conjunction with the SPI exact inference algorithm

immature Bayesian network when many variables in the model and their interrelationships have yet to be defined.

Because I am using the Boyen-Koller inference algorithm in conjunction with the SPI algorithm, the Average CPT Size factor is also a key factor in predicting the speed of inference of this algorithm. This factor is highly correlated to the average time to perform inference using the Boyen-Koller inference algorithm with a correlation coefficient over 0.9. This is to be expected as the speed of inference for the SPI algorithm should be strongly related to the average CPT size of the Bayesian network this inference algorithm is being used on. The reason for this is that the best the SPI algorithm can possibly do in terms of speed of inference is directly proportional to the total CPT size of the Bayesian network, or the product of the average CPT size and the number of nodes. If the SPI algorithm is able to marginalize out each node immediately after coming to that node in the factor tree, the variable that node represents would only have to be used in the equation once. Thus, the only variables required during each multiplication step would be the variables incorporated in the CPT table of the node being visited at that step. This lower limit in speed of inference is only possible on the most sparse of PDBNs and only if the optimal factor tree is found for the PDBN. However, the high correlation between the total CPT size of a PDBN and the speed of inference when using the Boyen-Koller inference algorithm over SPI seems to suggest that this lower limit in speed is often near the actual speed of inference for most PDBNs used in this study.

I also included the Number of Nodes factor when performing regression because the total CPT size of a Bayesian network is also highly correlated to how quickly the Boyen-Koller inference algorithm can perform inference on that network. Since the total CPT size of a Bayesian network is the product of that network's average CPT size and the number of nodes in that network, the Number of Nodes factor may also be a significant predictor of speed of inference for the Boyen-Koller inference algorithm over SPI.

It is important to note that the following models developed to predict the speed and accuracy of the Boyen-Koller inference algorithm over SPI also serve to model the speed and accuracy of the SPI exact inference algorithm. The Boyen-Koller inference algorithm over SPI is essentially the SPI algorithm when a single large cluster is used. Thus, one could consider the SPI algorithm the same algorithm as the Boyen-Koller inference algorithm when there is a single cluster containing all the nodes in the past expression. I took advantage of this fact when modeling for the Boyen-Koller inference algorithm by assuming the SPI algorithm was simply the Boyen-Koller inference algorithm at a fourth, slower speed emphasis level.

The first inference algorithm I examined was an exponential model of the speed of inference for the Boyen-Koller inference algorithm. The first set of factors I performed regression with was:

- Average States per Node
- Average CPT Size
- Number of Nodes
- Average Cluster Size

This model had a clear pattern in the residuals graph for the Average CPT Size factor. This graph can be seen in Figure 58 in Appendix C. The pattern appears to suggest a logarithmic relationship between the average CPT size of the PDBN and the natural logarithm of the speed of inference for the particle filter inference algorithm.

The Average States per Node factor was also not significant in this model. However, this may be due to potential logarithmic relationship between the Average CPT Size factor and the dependent variable. Thus, the set of factors I performed regression over in the next model were:

- Average States per Node
- Natural Logarithm of the Average CPT Size
- Number of Nodes
- Average Cluster Size

In this model, all factors including the Average States per Node factor were significant. None of the residual graphs for any of the factors listed above displayed any apparent patterns as seen in Figure 59, Figure 60, Figure 61, and Figure 62 in Appendix C.

Also, the graph of the residuals versus the natural logarithm of the Average CPT Size factor in this model does not appear to show a pattern as the Average CPT Size factor did in the previous model.

Figure 63 in Appendix C shows the model's fit to the actual time results while Figure 64 in Appendix C shows this same graph in logarithmic scale. Figure 65 in Appendix C shows the residual plot for this exponential model. From the graphs in Figure 63 through Figure 65, this exponential model appears to be a decent predictor of the actual speed of

inference for the particle filter inference algorithm. However, the residual plot shown in Figure 65 bring to question whether this model support the Gauss-Markov assumption that the error of the model is normal. It is hard to discern any particular pattern from any of these graphs. Nonetheless, this model may still be a decent predictor of the speed of inference for the Boyen-Koller algorithm based on these models.

A Normal Q-Q plot of the residuals, as seen in Figure 66 in Appendix C, shows that there are outliers at the left tail and right tails of the distribution. The right tail of the distribution appears to much heavier than the left tail of the distribution. The Q-Q plot along with the residual and model fit graphs above in Figure 59 through Figure 66 do not support the Gauss-Markov assumptions as clearly as in the best linear model above predicting the speed of inference of the particle filter inference algorithm. The model's error is not likely to be normal as seen from the Q-Q plot, although, the mean of the error appears to be near zero. However, the random errors of the model are all likely to be uncorrelated as no factor has a variance inflation factor above 1.3. Also, the lack of any noticeable patterns in any of the residual graphs for each of the independent variables suggests that random errors of the independent variables of the model are homoscedastic. However, this cannot be said of the dependent variable as seen in the residual graph for the dependent variable shown in Figure 65.

For the linear model of speed of inference for the Boyen-Koller inference algorithm over SPI, the factors I used were:

- Average States Per Node
- Average CPT Size

- Number of Static Nodes
- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Average Cluster Size

As usual, the Number of Dynamic Transitional Nodes and Average CPT Size factors were slightly correlated with a correlation of 0.321. Of greater concern was the fact that the Average States per Node factor had a negative coefficient when one would expect a positive coefficient. This model suggests that if the average states per node in a PDBN was large enough in value relative to the other factors used in this model listed above, that the time to perform inference with the Boyen-Koller inference algorithm could be negative.

The fact that there exist negative coefficients when one would expect a positive coefficient suggests that two or more factors or combinations of factors may be collinear. However, the largest variance inflation factor among the independent variables used in this model was 1.22, which is well below the threshold of 10.

In one attempt to generate a linear model without any negative coefficients, I combined each of the node class factors into the Number of Nodes factor. Thus, the factors I used in my second linear model were:

- Average States Per Node
- Average CPT Size
- Number of Nodes
- Average Cluster Size

Combining each of the node class factors into the Number of Nodes factor did not help, however. The Average States per Node factor still had a negative coefficient.

Furthermore, the Number of Nodes factor was insignificant in this model.

In my third linear model predicting the speed of inference of the Boyen-Koller inference algorithm and my second attempt to produce a linear model without any negative coefficients, I used the factors:

- Average States Per Node
- Average CPT Size
- Average Cluster Size

This model also had a negative coefficient for the Average States per Node factor, and thus, was not an adequate predictor of the dependent variable. The problem with a negative coefficient for the Average States per Node factor is that the model could potentially predict a negative speed of inference for some possible PDBNs and inference scenarios. Such a PDBN would likely be a sparse network such that the average states per node of that PDBN was much larger than the average CPT size of that same PDBN. Also, the fully factored Boyen-Koller inference algorithm would likely be used in this case.

In a final attempt to generate a linear model without a negative coefficient, I removed the Average States per Node factor that consistently came up with a negative coefficient.

This model is a modification of the original linear model where I used the factors:

- Average CPT Size
- Number of Static Nodes

- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Average Cluster Size

This model had no negative coefficients for any of the factors listed above. Furthermore, there were no apparent patterns in any of the residual plots of any of the independent variables.

However, the Number of Dynamic Non-Transitional Nodes factor was found to not be significant. Thus, a final linear model was generated where this factor was removed, making the list of factors in this model:

- Average CPT Size
- Number of Static Nodes
- Number of Dynamic Transitional Nodes
- Average Cluster Size

In this model, all factors were significant. None of the residual graphs for any of the factors listed above displayed any apparent patterns as seen in Figure 67, Figure 68, Figure 69, and Figure 70 in Appendix C. Each of these residual graphs show a few outliers. However, the largest Cook's distance value was 0.131, and thus, these outliers were likely not overly influential on the results of the regression.

Figure 71 in Appendix C shows the model's fit to the actual time results while Figure 72 in Appendix C shows the residual plot for this linear model. From the graphs in Figure 71 and Figure 72, this model appears to be a decent predictor of the actual speed of inference for the particle filter inference algorithm. However, like the exponential

model earlier, the residual plot shown in Figure 72 bring to question whether this model support the Gauss-Markov assumption that the error of the model is normal. As before, it is hard to discern any particular pattern from any of these graphs. Nonetheless, this model may also be a decent predictor of the speed of inference for the Boyen-Koller algorithm based on these models.

A Normal Q-Q plot of the residuals, as seen in Figure 73 in Appendix C, shows that there are outliers at the left and right tails of the distribution. The right tail of the distribution appears to be much heavier than the left tail of the distribution. The Q-Q plot along with the residual and model fit graphs above in Figure 67 through Figure 73 do not support the Gauss-Markov assumptions as clearly as in the best linear model above predicting the speed of inference of the particle filter inference algorithm. The model's error is not likely to be normal as seen from the Q-Q plot, although, the mean of the error is near zero. However, the random errors of the model are all likely to be uncorrelated as no factor has a variance inflation factor above 1.2. Also, the lack of any noticeable patterns in any of the residual graphs for each of the independent variables suggests that random errors of the independent variables model are homoscedastic. However, this cannot be said of the dependent variable as seen in the residual graph for the dependent variable shown in Figure 72.

3.5 Analysis and Results for the Boyen-Koller Inference Algorithm Variance of Speed of Inference

I used the same set of factors when developing both the speed of inference and variance of the speed of inference for the Boyen-Koller inference algorithm over SPI.

These factors were:

- Average States per Node
- Average CPT Size
- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Number of Static Nodes
- Number of Particles

Unlike the relationship between the speed of inference and the variance of the speed of inference for the particle filter algorithm, the relationship between these two metrics was much more complex for the Boyen-Koller inference algorithm. I graph the relationship between these two factors in Figure 9. From this graph, there appears to be an evident separation of values such that the dataset is likely to be covering two separate phenomenon.

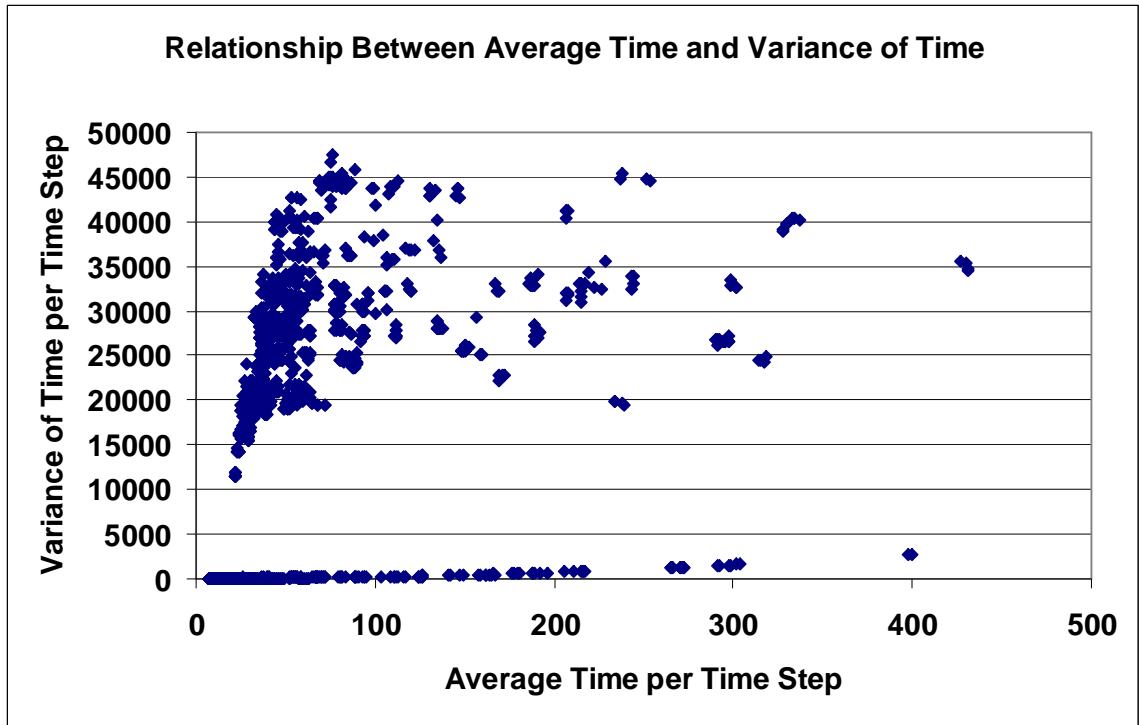


Figure 9: Relationship Between Average Time per Time Step and Variance of Time per Time Step for the Boyen-Koller Inference Algorithm

Upon further examination, I found the separation to be between the case where user defined clusters are used with the Boyen-Koller algorithm and the case where either the SPI algorithm is used alone or the Boyen-Koller algorithm uses fully factored clustering. Figure 10 shows the relationship between the average speed of inference and the variance of the speed of inference for the case when the user defines the clusters used by the Boyen-Koller inference algorithm. Figure 11 shows the relationship between the average speed of inference and the variance of the speed of inference for the SPI algorithm along with the Boyen-Koller algorithm when fully factored clustering is used. In both cases there appears to be a relationship between the speed of inference and the variance of the speed of inference in the given scenario.

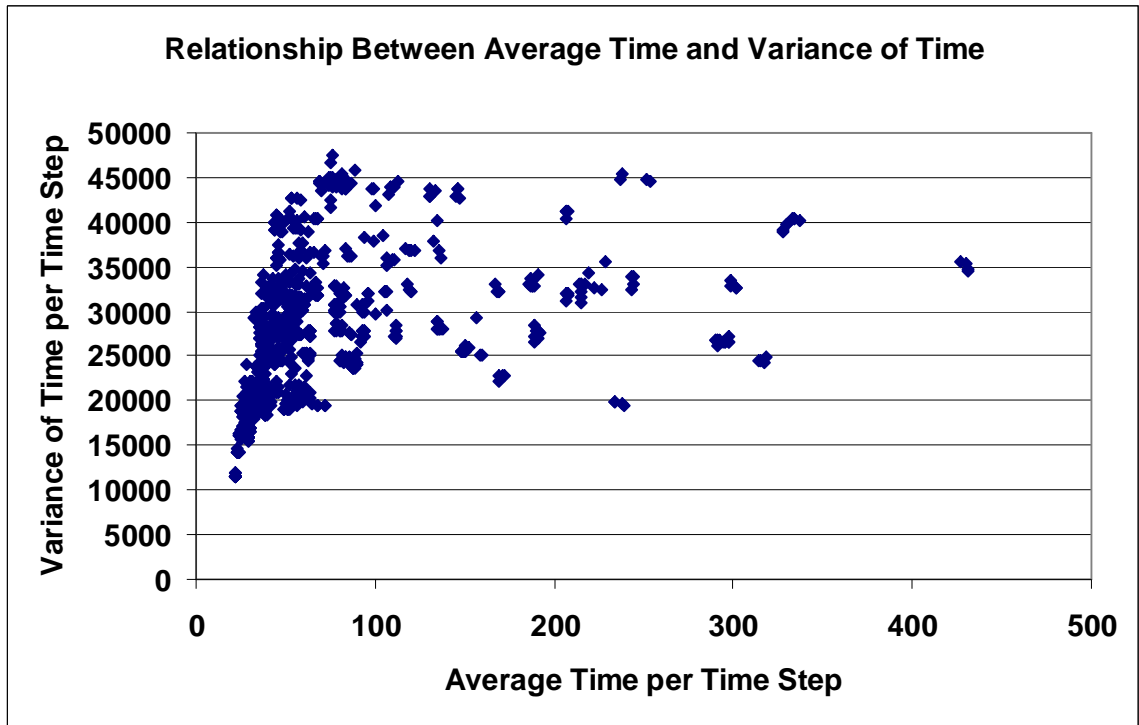


Figure 10: Relationship Between Average Time per Time Step and Variance of Time per Time Step for the Boyen-Koller Inference Algorithm with User Defined Clusters

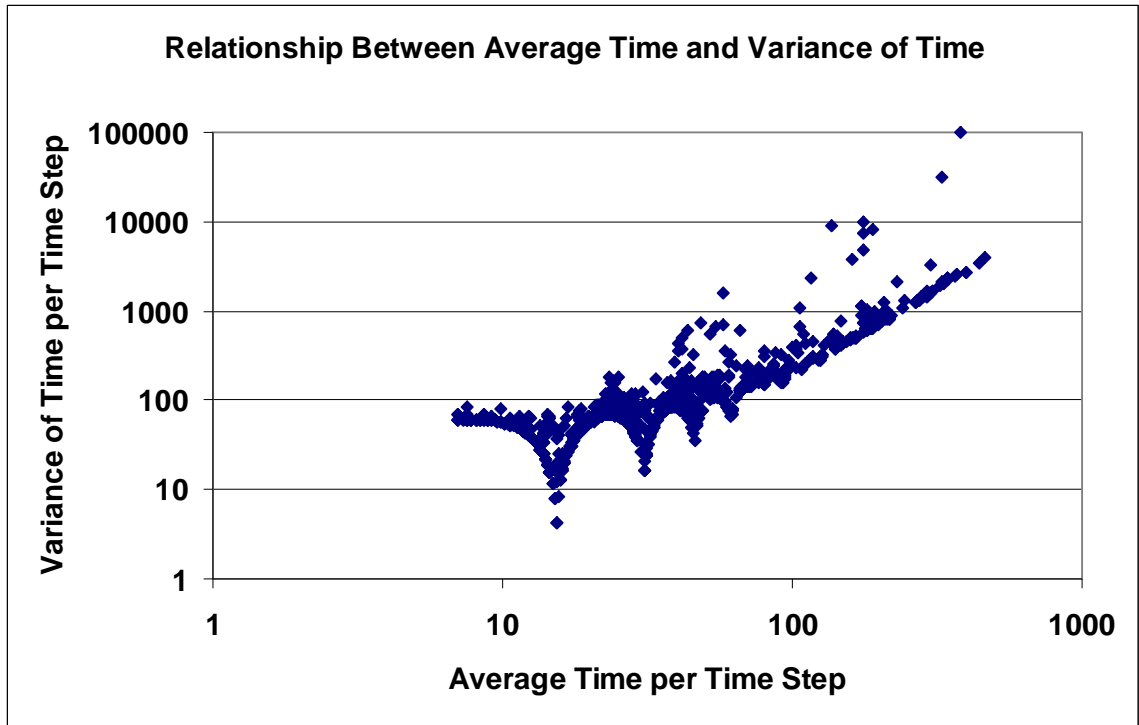


Figure 11: Relationship Between Average Time per Time Step and Variance of Time per Time Step for the Fully Factored Boyen-Koller Inference Algorithm and the SPI Algorithm

I believe that these relationships entail that the systems behind the speed of inference and the variance of the speed of inference for both these scenarios are similar. Thus, I believe the models that predict these two metrics in these two scenarios would likely be equally similar as well as the factors involved.

The first case analyzed was where the user selects the clusters to be used by the Boyen-Koller inference algorithm. The first inference algorithm I examined within this case was an exponential model of the variance of the speed of inference for the Boyen-Koller inference algorithm. The first set of factors I performed regression with was:

- Average States per Node
- Average CPT Size

- Number of Nodes
- Average Cluster Size

This model appears to have a pattern in the residuals graph for the Average CPT Size factor. This graph can be seen in Figure 74 in Appendix D. The pattern appears to suggest a logarithmic relationship between the average CPT size of a PDBN and the natural logarithm of the variance of the speed of inference for the Boyen-Koller inference algorithm.

There also is an apparent pattern in the residuals graph for the Average Cluster Size factor, although the nature of the pattern is less obvious. This graph can be seen in Figure 75 in Appendix D. The data points appear to be in two groupings in Figure 75 where the two groupings appear to have a near parallel slope to each other. Upon further examination, I found that the two groupings are partitioned by the two speed emphasis levels contained in this dataset. This dataset covers two of the four speed emphasis levels used for the Boyen-Koller inference algorithm: the slow speed emphasis level where 5 clusters are used and the medium speed emphasis level where 8 clusters are used.

The Number of Nodes factor was also not significant in this model. I performed regression again over the same set of factors excluding the Number of Nodes factor. Despite this change, there was still a clear pattern in the residuals graph for the Average CPT Size factor. The graph was nearly identical to the one seen in Figure 74. The same was true for the residuals graph for the Average Cluster Size factor shown in Figure 75.

The next exponential model I examined within this case used the natural logarithm of the average CPT size as a factor in response to the pattern seen in Figure 74 in the previous model. Thus, the set of factors I used with this model were:

- Average States per Node
- Natural Logarithm of the Average CPT Size
- Number of Nodes
- Average Cluster Size

The residuals graphs for each of these independent variables showed no apparent patterns with the exception of the Average Cluster Size factor. As in the previous model, the residuals graph for the Average Cluster Size factor appeared to have grouping of data points where each grouping had what appeared to be parallel positive slopes. The residual graphs for each of the independent variables can be seen in Figure 76 through Figure 79 in Appendix D.

As in the previous model, The Number of Nodes factor was not significant. I performed regression again over the same set of factors excluding the Number of Nodes factor. Despite this change, there was still a clear pattern in the residuals graph for the Average Cluster Size factor. The graph was nearly identical to the one seen in Figure 79.

Figure 80 in Appendix D shows the model's fit to the variance of the time results while Figure 81 in Appendix D shows this same graph in logarithmic scale. Figure 82 in Appendix D shows the residual plot for this exponential model. From the graphs in Figure 80 through Figure 82, this exponential model appears to be a poor predictor of the variance of the speed of inference for the Boyen-Koller inference algorithm over SPI.

The residual plot shown in Figure 82 has a clear pattern where the model error grows for larger values of variance.

A Normal Q-Q plot of the residuals, as seen in Figure 83 in Appendix D, shows that there are outliers at both tails of the distribution. The Q-Q plot along with the residual and model fit graphs above in Figure 76 through Figure 83 do not support several of the Gauss-Markov assumptions. The model's error is somewhat normal as seen from the Q-Q plot and appears to have a mean near zero. The random errors of the model are all likely to be uncorrelated as no factor has a variance inflation factor above 1.1. Also, the lack of any noticeable patterns in any of the residual graphs for each of the independent variables, with the exception of the Average Cluster Size factor, suggests that random errors of these independent variables of the model are homoscedastic. However, this cannot be said of the dependent variables as seen in the residuals graph for the dependent variable shown in Figure 82. Furthermore, the residuals graph for the Average Cluster Size factor shown in Figure 79 clearly does not have normal error.

Next, I examined various linear models of the variance of the speed of inference for the Boyen-Koller inference algorithm. The first set of factors I performed regression with was:

- Average States per Node
- Average CPT Size
- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Number of Static Nodes

- Average Cluster Size

Although this model had no apparent pattern in the residuals graphs for all the above independent variables, I did find that the Average CPT Size factor not to be significant.

Thus, in the next linear model I removed the Average CPT Size factor as an independent variable. The factors I performed regression over were then:

- Average States per Node
- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Number of Static Nodes
- Average Cluster Size

This model again had no apparent patten in the residual graphs for the independent variables used. However, I found that two new factors were not significant: the Number of Non-Transitional Nodes factor and the Average Cluster Size factor. Furthermore, the coefficients of many of the factors had changed significantly.

These results are indicative of multicollinearity existing in the model between two or more independent variables. Although in the last model the highest variance inflation factor for any of the independent variables was no higher than 1.4, there did exist a correlation between the Average Cluster Size factor and the Number of Dynamic Transitional Nodes factor with a correlation coefficient between the factors of approximately 0.35. Furthermore, there was a correlation between the Average CPT Size factor and the Number of Dynamic Transitional Nodes factor with a correlation coefficient between the factors of approximately 0.34. Despite the low values for the

variance inflation factors, I believe the present model had two or more factors that were collinear, and thus, disrupting the results of my regression analysis.

I decided to handle this multicollinearity by removing the Number of Dynamic Transitional Nodes factor that was correlated to two other independent variables in the first linear model. Thus, the factors used in this model were:

- Average States per Node
- Average CPT Size
- Number of Dynamic Non-Transitional Nodes
- Number of Static Nodes
- Average Cluster Size

Although removing this factor appeared to have removed the symptoms of multicollinearity I had seen in the previous model, the model that resulted was not a good predictor of the dependent variable.

Therefore, in the next linear model I instead removed the Average Cluster Size factor. The independent variables of this model were then:

- Average States per Node
- Average CPT Size
- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Number of Static Nodes

This model appears to have no pattern in the residuals graphs for each of the independent variables used. However, I found that the Average CPT Size factor was not significant.

Furthermore, this factor still was correlated to the Number of Dynamic Transitional Nodes factor.

Thus, in my final linear model for this particular case I removed the Average CPT Size factor leaving me with the following factors:

- Average States per Node
- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Number of Static Nodes

None of the residuals graphs for any of the independent variables had any apparent patterns as seen in Figure 84 through Figure 87 in Appendix D. I should also note that the coefficients of each factor in this model changed only slightly from their values in the subsequent model. Thus, this gives credence to the possibility that the correlation between the Average Cluster Size factor and the Number of Dynamic Transitional Nodes factor was what caused the change in coefficient values between models in the initial linear models I generated.

Figure 88 in Appendix D shows the model's fit to the variance of the time results while Figure 89 in Appendix D shows the residual plot for this linear model. From the graphs in Figure 88 and Figure 89, this linear model appears to be a decent predictor of the variance of the speed of inference for the Boyen-Koller inference algorithm over SPI. There appear to be a few outliers in the graphs shown in Figure 88 and Figure 89. However, the results of Cook's distance statistic suggest that none of these outliers were

overly influential on the results of the regression analysis. Also, it is hard to discern any particular pattern from either of these graphs.

A Normal Q-Q plot of the residuals, as seen in Figure 90 in Appendix D, shows that there are outliers at both tails of the distribution with the right tail of the distribution having the most dramatic outliers. The Q-Q plot along with the residual and model fit graphs above in Figure 84 through Figure 90 do not support several of the Gauss-Markov assumptions. The model's error is not normal as seen from the Q-Q plot although it appears to have a mean near zero. The random errors of the model are all likely to be uncorrelated as no factor has a variance inflation factor above 1.1. Also, the lack of any noticeable patterns in any of the residual graphs for each of the independent variables suggests that random errors of these independent variables of the model are homoscedastic. The dependent variable shown in Figure 82 also appears to be without any significant pattern, although, the error of the dependent variable appears to have a lower likelihood of being normal compared to the error for each of the independent variables.

The second case analyzed was where the exact inference SPI algorithm was used or the Boyen-Koller inference algorithm over SPI was used with fully factored clustering. The first inference algorithm I examined within this case was an exponential model of the variance of the speed of inference for the Boyen-Koller inference algorithm. The first set of factors I performed regression with was:

- Average States per Node
- Average CPT Size

- Number of Nodes
- Average Cluster Size

This model appears to have a pattern in the residuals graph for the Average CPT Size factor. This graph can be seen in Figure 91 in Appendix D. The pattern appears to suggest a logarithmic relationship between the average CPT size of a PDBN and the natural logarithm of the variance of the speed of inference for the Boyen-Koller inference algorithm.

The Average States per Node factor was also not significant in this model. I performed regression again over the same set of factors excluding the Average States per Node factor. Despite this change, there was still a clear pattern in the residuals graph for the Average CPT Size factor. The graph was nearly identical to the one seen in Figure 91.

For the next exponential I developed, I used the natural logarithm of the average CPT size in place of the Average CPT Size factor. Thus, the independent variables used in this model were:

- Average States per Node
- Natural Logarithm of the Average CPT Size
- Number of Nodes
- Average Cluster Size

None of the residuals graphs for any of the independent variables had any apparent patterns as seen in Figure 93 through Figure 95 in Appendix D. However, two of the factors, the Average States per Node factor and the Number of Nodes factor, were not

significant. I performed regression again over the same set of factors excluding the Average States per Node factor and the Number of Nodes factor.

Figure 96 in Appendix D shows the model's fit to the variance of the time results while Figure 97 in Appendix D shows this same graph in logarithmic scale. Figure 98 in Appendix D shows the residual plot for this exponential model. From the graphs in Figure 96 through Figure 98, this exponential model appears to be a poor predictor of the variance of the speed of inference for the Boyen-Koller inference algorithm over SPI. There are a number of outliers that are apparent in Figure 96, Figure 97, and Figure 98. Nonetheless, the results of Cook's distance statistic suggests that none of these outliers were overly influential on the results of the regression analysis. The residual plot shown in Figure 98 has a clear pattern where the model error grows for larger values of variance. This pattern appears logarithmic in nature.

A Normal Q-Q plot of the residuals, as seen in Figure 99 in Appendix D, shows that there are outliers at the right tails of the distribution. The Q-Q plot along with the residual and model fit graphs above in Figure 93 through Figure 99 do not support several of the Gauss-Markov assumptions. The model's error is not normal as seen from the Q-Q plot although it does appear to have a mean near zero. The random errors of the model are all likely to be uncorrelated as no factor has a variance inflation factor above 1.1. Also, the lack of any noticeable patterns in any of the residual graphs for each of the independent variables, with the exception of the Average Cluster Size factor, suggests that random errors of these independent variables of the model are homoscedastic. However, this cannot be said of the dependent variables as seen in the residuals graph for

the dependent variable shown in Figure 98. The dependent variable is clearly heteroscedastic where the relationship between the model's error and the dependent variable appears to be logarithmic.

Next, I examined various linear models of the variance of the speed of inference for the Boyen-Koller inference algorithm. The first set of factors I performed regression with was:

- Average States per Node
- Average CPT Size
- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Number of Static Nodes
- Average Cluster Size

There were several factors that I found not to be significant. These factors were the Average States per Node factor, the Number of Dynamic Transitional Nodes factor, and the Number of Static Nodes factor. I removed these factors and generated a new linear model using the factors:

- Average CPT Size
- Number of Dynamic Non-Transitional Nodes
- Average Cluster Size

This model appears to have a pattern in the residuals graph for the Average CPT Size factor where the nature of the pattern was either logarithmic or quadratic. The residuals graph for the Average CPT Size factor can be seen in Figure 100 in Appendix D. Thus, I

attempted new versions of the above model, replacing the Average CPT Size factor with the Natural Logarithm of the Average CPT Size factor and the Average CPT Size Square factor. Neither this model or the two alternative proved to be adequate models over the training dataset. In all cases, the R Square of the model was well below 0.3.

3.6 Analysis and Results for the Particle Filter Inference Algorithm Accuracy of Inference

For the accuracy of inference models for the particle filter inference algorithm, I used the factors:

- Average Node Variance
- Average States per Node
- Average CPT Size
- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Number of Static Nodes
- Number of Particles

The particle filter inference algorithm is a Monte Carlo method. One estimate of the error of the Monte Carlo method is:

$$\frac{\sigma(f)}{\sqrt{N}} \quad \text{Equation 6}$$

Where $\sigma(f)$ is the standard deviation of the function being sampled and N is the number of samples (Kuo 2005). Thus, it is likely that both the variance of the distribution of the

nodes being queried given evidence and the number of particles used when sampling are both important factors for the accuracy of the particle filter inference algorithm.

This error estimate may not be the only influencing component of the accuracy of the particle filter inference algorithm. Problems such as particle impoverishment may introduce other forms of error that may also affect the accuracy of this inference algorithm. Nonetheless, this error estimate is likely to be important in predicting the accuracy of the particle filter inference algorithm.

In the experiments I conducted, the node being queried was the static classifier node. I used the Average Node Variance factor as an approximation of the variance of the distribution of the classifier node given evidence. Since the average node variance is an average over each of the nodes in the Bayesian network, the number of each type of node was also included as a factor when developing this model. As with the previous models, I combined the three factors dealing with the number of each class of nodes in a PDBN when performing regression with the exponential model.

The Number of Static nodes alone is an important factor that may be significant in determining the accuracy of the particle filter inference algorithm. Static nodes or dynamic nodes that do not vary much from time step to time step can potentially cause the particle filter inference algorithm to prematurely converge to a particular set of node states. Here, premature means that the inference algorithm has become certain of the state of the system the PDBN is modeling yet this state could potentially be incorrect (Arulampalam 2002). Thus, static nodes within a PDBN can cause great difficulty for the particle filter inference algorithm to generate an accurate query result.

The remaining factors, the Average States per Node factor and the Average CPT Size factor, have no apparent significant relationship to the accuracy of the particle filter inference algorithm. I added these factors in later models when the factors described above proved inadequate in predicting the accuracy of the particle filter inference algorithm.

As before, I developed both linear and exponential models that predict the accuracy of the particle filter inference algorithm. In each model I developed, the R Square for the training set never exceeded 0.1 regardless of whether the model was linear or exponential.

The first model I examined was an exponential model of the accuracy of the particle filter inference algorithm. The set of factors used in this model was:

- Average Node Variance
- Number of Nodes
- Number of Particles

Here, the R Square for the training set used was 0.04. The only significant factors were the Average Node Variance factor and the Number of Particles factor.

One possible explanation for this low R Square value is due to certain significant factors not being included in the regression analysis to develop this model. Therefore, in the next exponential model I examined I included the Average States per Node and the Average CPT Size factors. Thus, the set of factors used in this model were:

- Average Node Variance
- Average States per Node

- Average CPT Size
- Number of Nodes
- Number of Particles

I know of no theoretical relationship between the Average States per Node and Average CPT size factors and the accuracy of the particle filter inference algorithm. However, these two factors represent two attributes of a PDBN that were not appreciably covered by the set of factors used in the previous model and that are not correlated with each other or any of the other factors.

As a final attempt, I added the Average Strength of Dependency factor to the model. Again, I know of no theoretical relationship between this factor and the accuracy of the particle filter inference algorithm. However, this factor is not correlated with any of the other factors used in the model and represents one more attribute of a PDBN that is not yet covered.

The factors I used in this exponential model were:

- Average Strength of Dependency
- Average Node Variance
- Average States per Node
- Average CPT Size
- Number of Nodes
- Number of Particles

This model was similar to the previous model both in the R Square value and the coefficients of each factor. The Average Strength of Dependency factor was not significant, although it was close to being significant.

Despite the addition of these three factors, the R Square for the training set used was still very low at 0.04. Again, the only significant factors were the Average Node Variance factor and the Number of Particles factor.

Although neither of these models were very accurate at predicting the accuracy of the particle filter inference algorithm, each model was statistically significant with an F statistic below 0.05. Furthermore, the same two factors were significant factors in each model:

- Average Node Variance
- Number of Particles

Also, these two factors had roughly the same coefficient values in each model. The coefficient for the Average Node Variance factor was approximately 3 and the coefficient for the Number of Particles factor was approximately -4×10^{-5} . Finally, the signs for the coefficients of these two factors matched what would be expected from Equation 6. The sign of the Average Node Variance factor was positive. The sign of the Number of Particles factor was negative signifying an inverse relationship between the number of particles used to sample the PDBN and the error in the inference results.

For the first linear model to predict the accuracy of the particle filter inference algorithm I used the factors:

- Average Node Variance

- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Number of Static Nodes
- Number of Particles

The R Square of this model for the training set used was 0.05. The only significant factor was the Number of Particles factor, although, the Average Node Variance factor had a P-value that nearly put it beyond the 0.05 threshold of significance.

As with the exponential models, I added the Average States per Node and Average CPT Size factors in the next model I developed in the unlikely circumstance these factors were significant and would make the model more accurate than the previous linear model. Thus, the factors I used in the next linear model were:

- Average Node Variance
- Average States per Node
- Average CPT Size
- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Number of Static Nodes
- Number of Particles

It is important to note that the Number of Dynamic Transitional Nodes and Average CPT Size factors were correlated to each other with a correlation coefficient of 0.32. This correlation is not very high and only just beyond my threshold for a correlation between

two factors. Nonetheless, the relationship between these two factors may have caused problems for the regression analysis.

The R-Square value for this model was only 0.06. The only significant factors were the Average States per Node and Number of Particles factors. Unlike the previous four models above, the Average Node Variance factor was not significant for this model.

For the last linear model, I tried adding the Strength of Dependency factor again. Thus, the factors used in this model were:

- Average Strength of Dependency
- Average Node Variance
- Average States per Node
- Average CPT Size
- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Number of Static Nodes
- Number of Particles

This model was nearly identical to the previous model both in the R Square value and the coefficients of each of the factors. Also, the Average Strength of Dependency factor was not significant.

3.7 Analysis and Results for the Boyen-Koller Inference Algorithm Accuracy of Inference

For the accuracy of inference models for the Boyen-Koller inference algorithm over SPI, I used the factors:

- Average Strength of Dependency
- Average States per Node
- Average CPT Size
- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Number of Static Nodes
- Average Cluster Size

The Boyen-Koller inference algorithm is essentially removing arcs from the Bayesian network to simplify the complexity of inference. The effective removal of these arcs in the past expression increases the speed of inference but also reduce the accuracy of the inference. The amount of error in the results is related to the number of arcs removed and the strength of the relationship between the nodes those arcs connected. I used the Average Strength of Dependency factor as a proxy measure of the arcs removed by the Boyen-Koller inference algorithm.

The average cluster size used when performing inference with the Boyen-Koller inference algorithm over SPI is also a factor that is likely to affect the accuracy of inference of this algorithm. The smaller and more numerous the clusters used by the

Boyen-Koller inference algorithm, the more arcs are removed and the less accurate the inference algorithm.

Finally, the number of static nodes is likely to be another important factor in predicting the accuracy of the Boyen-Koller inference algorithm. As in the particle filter inference algorithm, the existence of static nodes or dynamic nodes that do not vary much from time step to time step can reduce the accuracy of inference for the inference algorithm. In this case, the bound on the error when using the Boyen-Koller inference algorithm can no longer be guaranteed when the dynamic Bayesian network contains static nodes.

The remaining factors have no apparent significant relationship to the accuracy of the particle filter inference algorithm. I added these factors in later models when the factors described above proved inadequate in predicting the accuracy of the particle filter inference algorithm. These factors are the Average States per Node, the Average CPT Size, the Number of Dynamic Transitional Nodes, and the Number of Dynamic Non-Transitional Nodes.

As before, I developed both linear and exponential models that predict the accuracy of Boyen-Koller inference algorithm. In each model I developed, the R Square for the training set never exceeded 0.1 regardless of whether the model was linear or exponential.

The first model I examined was an exponential model of the accuracy of the Boyen-Koller inference algorithm over SPI. The set of factors used in this model was:

- Average Strength of Dependency

- Number of Static Nodes
- Average Cluster Size

Here, the R Square for the training set used was 0.008. There were no significant factors and the model was not significant.

In the next model I included the Average States per Node and Average CPT Size factors as well as replace the Number of Static Nodes factor with the Number of Nodes factor. As with the particle filter accuracy models, I added the Average States per Node and Average CPT Size factors with the hope that these factors may be significant. I replaced the Number of Static Nodes factor with the Number of Nodes factor so the other types of nodes in a PDBN would be represented along with the number of static nodes in a PDBN. Again, this was to see if these extra factors which are components of the Number of Nodes factor may be significant. The set of factors in this next exponential model was then:

- Average Strength of Dependency
- Average States per Node
- Average CPT Size
- Number of Nodes
- Average Cluster Size

Despite the addition of these factors, the R Square for the training set used was still very low at 0.02. The only significant factor here was the Average Cluster Size factor, although, the Number of Nodes factor was near the threshold of significance with a P-

value of 0.51. The Average Cluster Size factor was positive while the Number of Nodes factor was negative. The model was significant this time.

For the first linear model to predict the accuracy of the Boyen-Koller inference algorithm I used the factors:

- Average Strength of Dependency
- Number of Static Nodes
- Average Cluster Size

The R Square for this model was 0.02 and the model was significant. Both the Number of Static Nodes and the Average Cluster Size factors were significant.

I then added the Average States per Node, the Average CPT Size, the Number of Dynamic Transitional, and the Number of Dynamic Non-Transitional Nodes factors in the unlikely circumstance these factors were significant. Thus, the factors I used in the next linear model were:

- Average Strength of Dependency
- Average States per Node
- Average CPT Size
- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Number of Static Nodes
- Average Cluster Size

It is important to note that the Number of Dynamic Transitional Nodes and Average CPT Size factors were correlated to each other with a correlation coefficient of 0.32. This

correlation is not very high and only just beyond my threshold of 0.3 for a correlation between two factors. Nonetheless, the relationship between these two factors may have caused problems for the regression analysis.

The R Square for this model was 0.03 and the model was significant. The only significant factors were the Number of Static Nodes and the Average Cluster Size factors.

4. Conclusions

4.1 Discussion of the Time Models

From my research I was able to determine what factors in Table 1 have a statistically significant effect on each of the inference algorithms examined in this study. I was also able to develop statistical models, which show the relationship between these factors and the inference algorithms I examined.

Of all the models I developed predicting the speed of inference for the particle filter inference algorithm, the two best are the fourth exponential model using the factors:

- Natural Logarithm of the Average CPT Size
- Number of Nodes
- Natural Logarithm of the Number of Particles

And the first linear model using the factors:

- Average States Per Node
- Average CPT Size
- Number of Static Nodes
- Number of Dynamic Transitional Nodes
- Number of Dynamic Non-Transitional Nodes
- Number of Particles

Table 8 lists the coefficients for each of the factors used in the exponential model as well as statistics on the significance of each of those factors. Table 9 lists the regression statistics associated with the linear regression model for the natural logarithm of the average time per time step for the particle filter inference algorithm. The final statistic in Table 9, the “R square for the test set” statistic, is the R square statistic when comparing the model’s predicted value against the actual speed of inference values over the entire test set. The previous 3 statistics listed in Table 9 are the statistics collected when using the training set. The training set is the data set used in the regression while the test set is the entire data set containing the training set as well as the holdout set.

Table 8: Particle Filter Average Speed Exponential Regression Model Statistics

	Coefficients	Standard Error	t Stat	P-value
Intercept	-3.239	0.114	-28.519	< 0.001
Average States Per Node	0.052	0.017	3.097	0.002
Natural Logarithm of the Average CPT Size	0.166	0.006	26.227	< 0.001
Number of Nodes	0.095	0.006	17.116	< 0.001
Natural Logarithm of the Number of Particles	0.857	0.006	143.309	< 0.001

Table 9: Particle Filter Average Speed Exponential Regression Model Regression Statistics

R Square	0.968555121
Adjusted R Square	0.96837994
Observations	723
R Square for Test Set	0.962345

Table 10 lists the coefficients for each of the factors used in the linear model as well as statistics on the significance of each of those factors. Table 11 lists the regression statistics associated with the linear regression model for the average time per time step

for the particle filter inference algorithm. As before, the final statistic in Table 11, the “R square for the test set” statistic, is the R square statistic when comparing the model’s predicted value against the actual speed of inference values over the entire test set. Also, the first three statistics in Table 11 are derived using the training set only while the final statistic in Table 11 is derived using the full test set.

Table 10: Particle Filter Average Speed Linear Regression Model Statistics

	Coefficients	Standard Error	t Stat	P-value
Intercept	-1112.288	43.096	-25.809	< 0.001
Average States Per Node	50.281	6.749	7.450	< 0.001
Average CPT Size	0.339	0.011	31.054	< 0.001
Number of Dynamic NonTransitional Nodes	78.492	2.900	27.063	< 0.001
Number of Dynamic Transitional Nodes	108.925	3.133	34.766	< 0.001
Number of Static Nodes	15.496	4.622	3.352	< 0.001
Number of Particles	0.087	0.000	184.952	< 0.001

Table 11: Particle Filter Average Speed Linear Regression Model Regression Statistics

R Square	0.981590458
Adjusted R Square	0.981437471
Observations	729
R Square for Test Set	0.980544

Although the linear model had two marginally correlated factors and the residuals plot for the dependent variable showed a logarithmic or quadratic pattern in it, this model still had a very good fit to the actual test data set as seen in Figure 32. The linear model also had a higher R square value for the test data set than the exponential model had. Furthermore, the exponential model appeared to have error that would grow linearly as the dependent variable grew linearly as shown in Figure 22. Thus, I believe the first

linear model above is the best predictor of average speed of inference per time step for the particle filter inference algorithm. It is important to note that this conclusion does not agree with the theoretical computational complexity of the particle filter inference algorithm. One would expect that the speed of inference for the particle filter inference algorithm would be the product of the number of nodes in the PDBN and the number of particles used in inference rather than the sum of the two.

Of all the models I developed predicting the variance of the speed of inference for the particle filter inference algorithm, the best was the third exponential model using the factors:

- Average States per Node
- Natural Logarithm of the Average CPT Size
- Number of Nodes
- Natural Logarithm of the Number of Particles

Table 12 lists the coefficients for each of the factors used in the exponential model as well as statistics on the significance of each of those factors. Table 13 lists the regression statistics associated with the linear regression model for the natural logarithm of the variance of the average time per time step for the particle filter inference algorithm. As before, the first three statistics in Table 13 are derived using the training set only while the final statistic in Table 13 is derived using the full test set.

Table 12: Particle Filter Variance of the Speed Exponential Regression Model Statistics

	Coefficients	Standard Error	t Stat	P-value
Intercept	-11.479	0.428	-26.809	< 0.001
Average States Per Node	0.272	0.062	4.389	< 0.001
Natural Logarithm of the Average CPT Size	0.139	0.024	5.780	< 0.001
Number of Nodes	0.332	0.021	15.930	< 0.001
Natural Logarithm of the Number of Particles	1.990	0.022	90.057	< 0.001

Table 13: Particle Filter Variance of the Speed Exponential Regression Model Regression Statistics

R Square	0.922037593
Adjusted R Square	0.921604469
Observations	725
R Square for Test Set	0.771123

Of all the models I developed predicting the speed of inference for the Boyen-Koller inference algorithm over SPI, the two best are the second exponential model using the factors:

- Average States per Node
- Natural Logarithm of the Average CPT Size
- Number of Nodes
- Average Cluster Size

And the fifth linear model using the factors:

- Average CPT Size
- Number of Static Nodes
- Number of Dynamic Transitional Nodes
- Average Cluster Size

Table 14 lists the coefficients for each of the factors used in the exponential model as well as statistics on the significance of each of those factors. Table 15 lists the regression statistics associated with the linear regression model for the natural logarithm of the average time per time step for the Boyen-Koller inference algorithm over SPI. As before, the first three statistics in Table 15 are derived using the training set only while the final statistic in Table 15 is derived using the full test set.

Table 14: Boyen-Koller Average Speed Exponential Regression Model Statistics

	Coefficients	Standard Error	t Stat	P-value
Intercept	-0.046	0.165	-0.278	0.781
Average States Per Node	-0.276	0.028	-9.860	< 0.001
Natural Logarithm of the Average CPT Size	1.003	0.011	93.542	< 0.001
Number of Nodes	0.095	0.009	10.507	< 0.001
Average Cluster Size	0.031	0.002	17.823	< 0.001

Table 15: Boyen-Koller Average Speed Exponential Regression Model Regression Statistics

R Square	0.923351836
Adjusted R Square	0.9230274
Observations	950
R Square for Test Set	0.881465

Table 16 lists the coefficients for each of the factors used in the linear model as well as statistics on the significance of each of those factors. Table 17 lists the regression statistics associated with the linear regression model for the average time per time step for the Boyen-Koller inference algorithm.

Table 16: Boyen-Koller Average Speed Linear Regression Model Statistics

	Coefficients	Standard Error	t Stat	P-value
Intercept	-205.392	21.331	-9.629	< 0.001
Average CPT Size	0.944	0.012	75.927	< 0.001
Number of Dynamic Transitional Nodes	25.002	3.351	7.460	< 0.001
Number of Static Nodes	27.949	4.986	5.605	< 0.001
Average Cluster Size	6.254	0.474	13.192	< 0.001

Table 17: Boyen-Koller Average Speed Linear Regression Model Regression Statistics

R Square	0.883295129
Adjusted R Square	0.88280114
Observations	950
R Square for Test Set	0.876233

Both these linear and the exponential models for the Boyen-Koller inference algorithm over SPI have problems with heteroscedastic. These two models also do not appear to have normal error. However, both these models still are able to predict the time to perform inference accurately over the test set with R Squares above 0.85 in both cases. I believe the exponential model is a better predictor of the speed of inference for the Boyen-Koller inference algorithm because it has the higher R Square for the test set. One would also expect an exponential model to predict the speed of inference of the Boyen-Koller inference algorithm as the theoretical computational complexity of this inference algorithm is also exponential. In fact, both the computational complexity model and the statistical model I developed have the Number of Nodes and the Average States per Node factor as significant factors.

However, it is important to note that the exponential model for the Boyen-Koller inference algorithm that I developed has a negative coefficient for the Average States per

Node factor. The negative coefficient implies an inverse relationship between the average states per node in a PDBN and the speed of inference on that PDBN when using the Boyen-Koller inference algorithm over SPI and when all other factors are taken into account. In contrast, the theoretical computational complexity of the Boyen-Koller inference algorithm shows a direct relationship between the speed of inference and the Average States per Node factor.

For the variance of speed of inference for the Boyen-Koller inference algorithm over SPI, I made two separate sets of models. The first set of models covered the case where the user specifies the clusters the algorithm would use. The second set of models covered the case where either the SPI algorithm alone was used or the Boyen-Koller inference algorithm was used with fully factored clustering.

Of all the models I developed predicting the variance of the speed of inference for the Boyen-Koller inference algorithm over SPI, the two best for the first case are the second exponential model using the factors:

- Average States per Node
- Natural Logarithm of the Average CPT Size
- Average Cluster Size

And the fourth linear model using the factors:

- Average States per Node
- Number of Dynamic Non-Transitional Nodes
- Number of Dynamic Transitional Nodes
- Number of Static Nodes

Table 18 lists the coefficients for each of the factors used in the exponential model as well as statistics on the significance of each of those factors. Table 19 lists the regression statistics associated with the linear regression model for the natural logarithm of the average time per time step for the Boyen-Koller inference algorithm over SPI. As before, the first three statistics in Table 19 are derived using the training set only while the final statistic in Table 19 is derived using the full test set.

Table 18: Boyen-Koller Variance of the Speed Exponential Regression Model Statistics

	Coefficients	Standard Error	t Stat	P-value
Intercept	-98717.241	1152.350	-85.666	< 0.001
Average States Per Node	9432.155	176.604	53.409	< 0.001
Number of Dynamic NonTransitional Nodes	5225.256	77.067	67.801	< 0.001
Number of Dynamic Transitional Nodes	9929.778	80.532	123.302	< 0.001
Number of Static Nodes	5329.108	121.696	43.790	< 0.001

Table 19: Boyen-Koller Variance of Speed Exponential Regression Model Regression Statistics

R Square	0.313231988
Adjusted R Square	0.308829629
Observations	472
R Square for Test Set	0.300021

Table 20 lists the coefficients for each of the factors used in the linear model as well as statistics on the significance of each of those factors. Table 21 lists the regression statistics associated with the linear regression model for the average time per time step for the Boyen-Koller inference algorithm.

Table 20: Boyen-Koller Variance of Speed Linear Regression Model Statistics

	Coefficients	Standard Error	t Stat	P-value
Intercept	-98717.241	1152.350	-85.666	< 0.001
Average States Per Node	9432.155	176.604	53.409	< 0.001
Number of Dynamic NonTransitional Nodes	5225.256	77.067	67.801	< 0.001
Number of Dynamic Transitional Nodes	9929.778	80.532	123.302	< 0.001
Number of Static Nodes	5329.108	121.696	43.790	< 0.001

Table 21: Boyen-Koller Variance of Speed Linear Regression Model Regression Statistics

R Square	0.97615874
Adjusted R Square	0.975950064
Observations	462
R Square for Test Set	0.970221

The exponential model for the Boyen-Koller inference algorithm over SPI has problems with heteroscedasticity. This model also does not appear to have normal error. Also, only the linear model is able to predict the variance of time to perform inference accurately over the test set with R Squares above 0.85. Thus, I believe the linear model is the better predictor of the variance of the speed of inference for the Boyen-Koller inference algorithm in the case where the uses defines the clusters to be used with the Boyen-Koller algorithm.

The best model for the case where either the Boyen-Koller inference algorithm is used with fully factored clustering or the SPI algorithm is used alone is the second exponential model using the factors:

- Natural Logarithm of the Average CPT Size
- Average Cluster Size

Table 22 lists the coefficients for each of the factors used in the model as well as statistics on the significance of each of those factors. Table 23 lists the regression statistics associated with the linear regression model for the natural logarithm of the average time per time step for the Boyen-Koller inference algorithm over SPI. As before, the first three statistics in Table 23 are derived using the training set only while the final statistic in Table 23 is derived using the full test set.

Table 22: Boyen-Koller Variance of the Speed Exponential Regression Model Statistics

	Coefficients	Standard Error	t Stat	P-value
Intercept	-0.481	0.190	-2.529	0.012
Natural Logarithm of the Average CPT Size	0.997	0.040	25.182	< 0.001
Average Cluster Size	0.101	0.010	10.378	< 0.001

Table 23: Boyen-Koller Variance of Speed Exponential Regression Model Regression Statistics

R Square	0.632288812
Adjusted R Square	0.630696989
Observations	465
R Square for Test Set	0.010807

When comparing the speed of inference of the particle filter inference algorithm and the Boyen-Koller inference algorithm over SPI, one key point to note is the fact that the speed of inference model for the Boyen-Koller inference algorithm is an exponential while the model for the particle filter inference algorithm is linear. From the empirical experiments, it was clear that the Boyen-Koller inference algorithm generally performed faster than the particle filter inference algorithm. However, the speed of inference models for both of these algorithms suggests that the time to perform inference with the

Boyen-Koller inference algorithm will grow much faster than the particle filter inference algorithm with a linear growth in PDBN complexity. Also, the particle filter inference algorithm is likely to be able to perform inference in a timely manner on PDBNs of a much higher complexity than the Boyen-Koller inference algorithm. Thus, although the Boyen-Koller inference algorithm is generally faster for relatively simple PDBNs, the particle filter inference algorithm has a much larger PDBN complexity range of useful speed of inference.

In the speed of inference models for each algorithm, the three significant factors shared by both models that appear to affect speed of inference are the Average States per Node factor, Average CPT Size factor, and the Number of Nodes factor. When comparing the two statistical models, the speed in which the Boyen-Koller inference algorithm can perform inference appears to be far less sensitive to the average number of states per node in the PDBN than the particle filter inference algorithm. In fact, the Boyen-Koller inference algorithm may actually run faster as the average number of states per node increases.

Conversely, the Boyen-Koller inference algorithm appears to be much more sensitive to the Average CPT Size. This is understandable as the particle filter inference algorithm is a Monte Carlo method, and thus, does not need to deal directly with the states of a node except to randomly select a hypothetical value for the node when generating particles. Nonetheless, the particle filter inference algorithm is still influence by the Average CPT Size of a PDBN since the number of particles used by the particle filter inference

algorithm must adequately cover the state space of the PDBN it is performing inference on if the algorithm is to produce accurate results.

The variance of the speed of inference of the particle filter inference algorithm is high relative to the average speed of inference for the algorithm. Among the test data set, the average variance of speed of inference was approximately 286,000 and the average percent variance of speed of inference ³was approximately 30,000%. Furthermore, the statistical model for the variance of the speed of inference for the particle filter inference algorithm is exponential. Therefore, as the complexity of the PDBN the inference is being performed on increases linearly, the variance in the time per time step for the particle filter inference algorithm to perform inference increases exponentially. Thus, even though the particle filter inference algorithm can perform inference on very large and complex PDBNs relative to the Boyen-Koller inference algorithm, the time per time step in which this algorithm performs inference can vary dramatically for these larger PDBNs.

When one defines the clusters to use with the Boyen-Koller inference algorithm over SPI, the variance of the speed of inference is also high relative to the average speed of inference for the algorithm. Among the test data set, the average variance of speed of inference was approximately 27,000 and the average percent variance of speed of inference was approximately 51,000%. However, unlike the particle filter inference algorithm, the statistical model for the variance of speed of inference for this inference algorithm is linear. Thus, the variance in the time to perform inference per time step for

³ The percent variance of speed of inference for a single test run of an inference algorithm was the percentage of the variance of the speed of inference relative to the speed of inference.

this algorithm will increase much less dramatically for large and complex PDBNs than the particle filter inference algorithm. However, the Boyen-Koller inference algorithm may still be unable to perform inference on these larger PDBNs in a timely manner.

Less can be said about the Boyen-Koller inference algorithm when it is fully factored or the Symbolic Probabilistic Inference algorithm. This is because the statistical model for the variance of the speed of inference for these inference algorithms in these situations was very inaccurate. From the empirical results, however, it appears that the variance of the speed of inference is low relative to the average speed of inference. Among the test data set, the average variance of speed of inference was approximately 393 and the average percent variance of speed of inference was approximately 390%. Thus, it appears that the SPI algorithm and the Boyen-Koller inference algorithm with fully factored particles have the least variability in the time to perform inference each time step.

4.2 Discussion of the Accuracy Models

I was unable to develop an accurate statistical model to predict the accuracy for any of the inference algorithms research in this study. I believe there are two likely reasons for this. Neither reason is mutually exclusive of the other. One possibility is that I did not use an adequate set of factors when developing the regression models to describe the accuracy of any of the inference algorithms. I had a great deal of difficulty finding any number of significant factors. This difficulty may be in part due to the fact that the scope of this research limited me to only use factors that could be easily collected by a

knowledge engineer and used before a Bayesian network has been fully defined. It is very possible that the key factors in predicting the accuracy of the inference algorithms I examined in this research study are complex factors that are outside of the scope of this research study. Another possibility is that the relationship between the factors I chose and the accuracy of each of the inference algorithms I examined is not linear or near linear. If this were the case, then simple linear regression may not be adequate to develop a useful model.

4.3 Future Work

Probably the biggest step one could take to improve upon this research would be to use distributions with wider ranges for the control variable distributions used by the Random PDBN Generator. The control variables used to randomly generate PDBNs used in this research study had significantly small variances as well as a very dramatic mode. The most notable example would be the Number of Static Nodes control variable that had an effective range of 2 to 4 static nodes, of which the vast majority of the generated PDBNs had three nodes. The effect of using these low variance distributions was that the sample set of PDBNs used in this research study were more homogeneous than they needed to be. This, in effect, weakened my statistical analysis.

Another weakness of this research was a lack of any good accuracy models for the inference algorithms studied. In the future, it may be beneficial to expand the scope of this study to cover any potential factor related to the architecture of a PDBN or the inference algorithm itself. In doing then, one could then develop factors that better

predict the accuracy of one or more inference algorithms. Some good candidate factors to use in future work include:

- Joint Network Variance – Rather than calculating the average variance of every node in the Bayesian network, a more informative factor might be to calculate the variance of the joint distribution of all the nodes in the PDBN. This variance could easily be calculated through Monte Carlo sampling much like how the Average Node Variance factor was approximated.
- Classifier Node Variance – It is the classifier node that we are querying at every time step as well as using to determine the accuracy of the approximate inference algorithms being studied. Therefore, it makes sense to calculate the variance of this node specifically to use as a factor in predicting the accuracy of the approximate inference algorithms. Again, this factor could easily be calculated through Monte Carlo sampling.
- Average Strength of Dependency between Intercluster Arcs – Rather than calculate the correlation of every node pair in the PDBN connected by an arc, it may be more useful to select only those connected node pairs that are between clusters used in the Boyen-Koller inference algorithm. The arcs that traverse between clusters are effectively removed by the Boyen-Koller inference algorithm. Therefore, specifically measuring the strength of the relationship these arcs represent may be beneficial in predicting the accuracy of the Boyen-Koller inference algorithm.

- Correlation of Nodes to Themselves between Time Steps – Static nodes, or any node that has a near deterministic change of its probability distribution between time steps to be more accurate, have the potential to decrease the accuracy of approximate inference algorithm. Therefore, calculating the degree to which nodes correlate with themselves on average between time steps may provide some insight into the accuracy of approximate inference algorithms.

Finally, future research could address some of the drawbacks of the random PDBN generator. The random PDBN generator was useful in creating a very large and varied sample set of PDBNs for use in this thesis research. However, the unsophisticated methods that the random PDBN generator utilized also created some problems the quality of the sample set. Because factors such as the number of arcs, the number of nodes, and the arrangement of arcs and nodes was chosen completely at random, a large number of large PDBNs generated by the random PDBN generator were too complex to be tractable with any of the inference algorithms used in this research. This resulted in wasted resources as well as potentially biasing my results.

Conversely, a knowledge engineer developing PDBNs of equivalent size and complexity would have a greater degree of success in developing tractable PDBNs. This is because a knowledge engineer is not blindly constructing the Bayesian network architecture, but rather, using various heuristics as well as instinct to optimize the efficiency in which a PDBN can run for a given PDBN complexity level. Thus, for a given preset PDBN size and complexity, a knowledge engineer will be able develop many more tractable PDBNs than a brute force random PDBN generator because the

knowledge engineer will be able to evaluate multiple alternatives towards the same end and optimize a PDBN using heuristics.

Another failing of the random PDBN generator is that it is unable to incorporate a great deal of domain specific attributes to the PDBNs it generates. A partially dynamic Bayesian network is a representation of knowledge within a specific problem domain. It is rarely developed without a specific problem in mind and the PDBN is always tailored specifically to that problem. Because of this, the domain in which a PDBN is being developed for is likely a significant factor in how what architecture that PDBN is likely to have. For example, the PDBNs from one domain may have a tendency to be sparse networks with many nodes while the PDBNs from another domain may generally be dense networks with few nodes. The random PDBN generator is able to provide a general picture of the full range of PDBNs possible. However, a knowledge engineer is likely only interested in a small subset of those possible PDBNs that are pertinent to his or her problem domain.

Thus, in future research it may make sense to use both randomly generated PDBNs as well as domain specific PDBNs generated by actual knowledge engineers. The randomly generated PDBNs would still provide breadth to the research allowing the researcher to examine a wide variety of PDBN architectures. At the same time, those domain specific PDBNs generated by knowledge engineers could provide depth to the research by providing samples from a specific domain.

Appendix A: Regression Diagnostics for the Particle Filter Speed of Inference Models

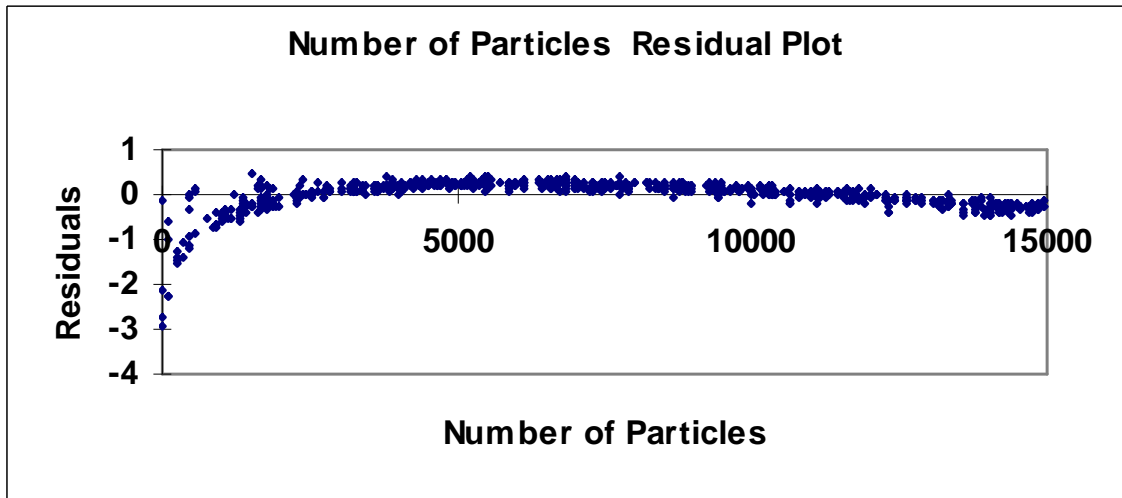


Figure 12: Particle Filter Average Speed Exponential Regression Model 1 Residuals versus Number of Particles Graph

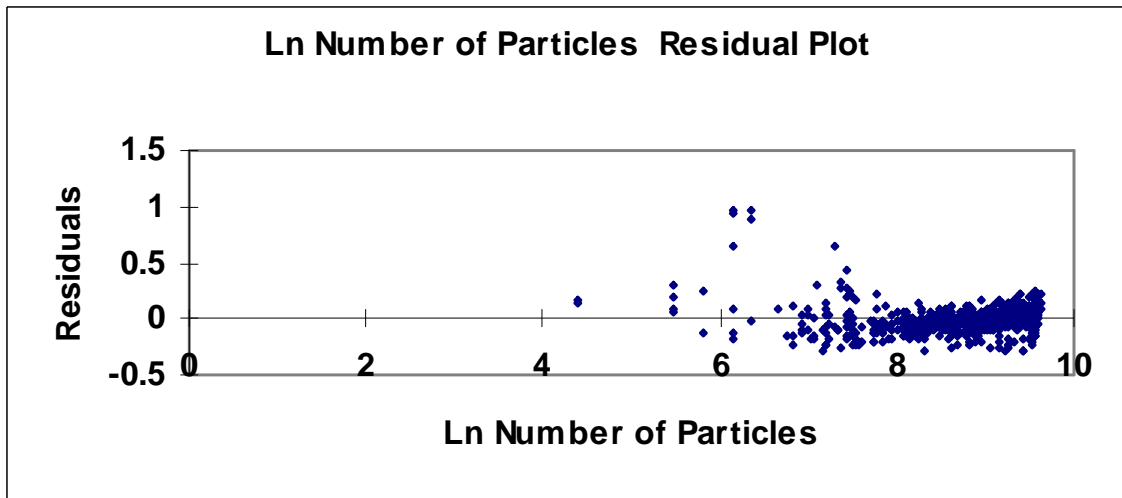


Figure 13: Particle Filter Average Speed Exponential Regression Model 2 Residuals versus Number of Particles Graph

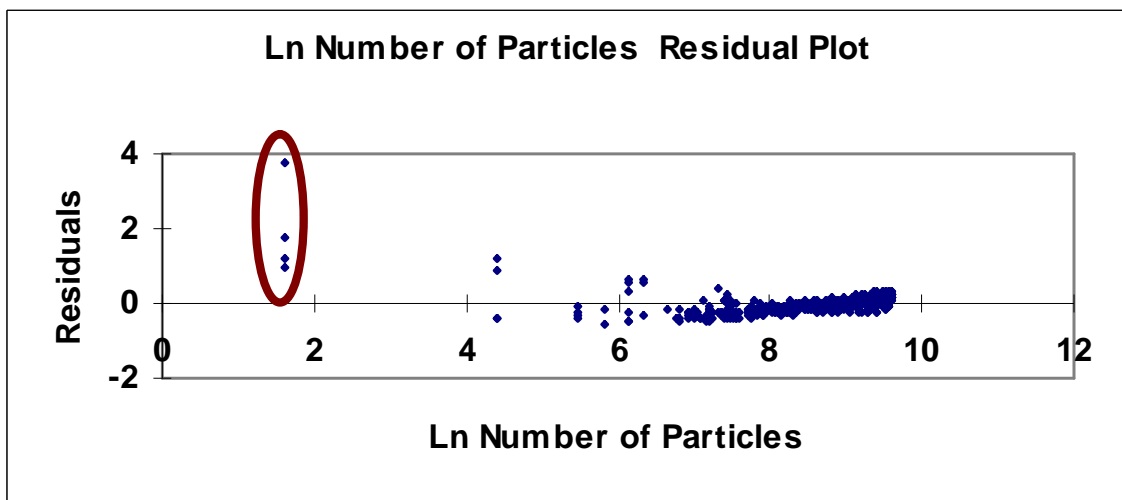


Figure 14: Particle Filter Average Speed Exponential Regression Model 2 Residuals versus Number of Particles Graph Before Removal of Outliers

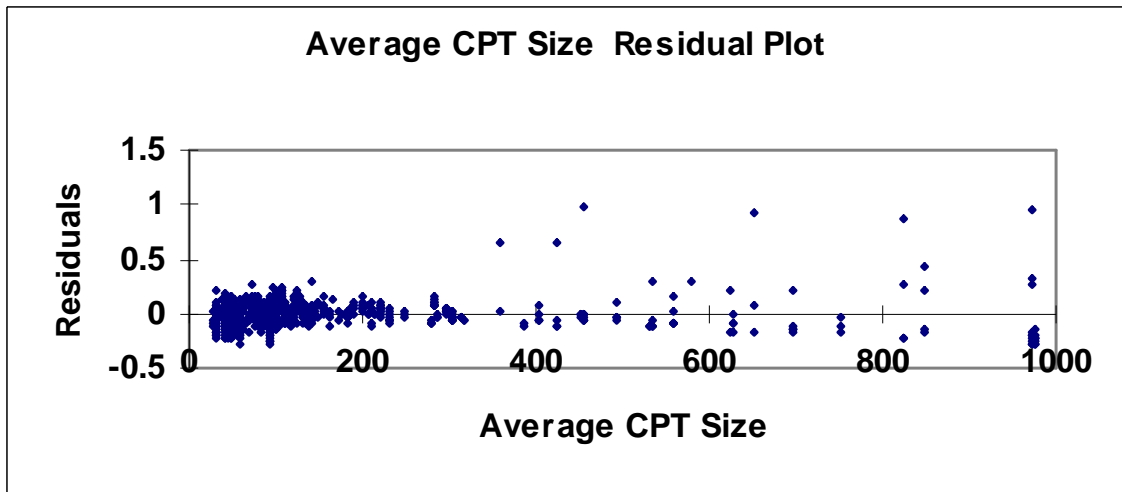


Figure 15: Particle Filter Average Speed Exponential Regression Model 2 Residuals versus Average CPT Size Graph

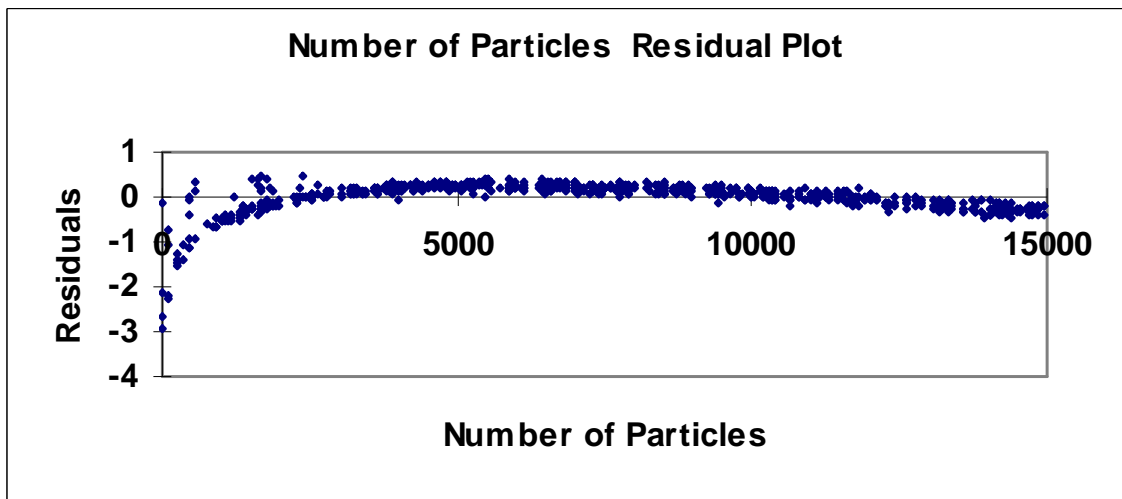


Figure 16: Particle Filter Average Speed Exponential Regression Model 3 Residuals versus Number of Particles Graph

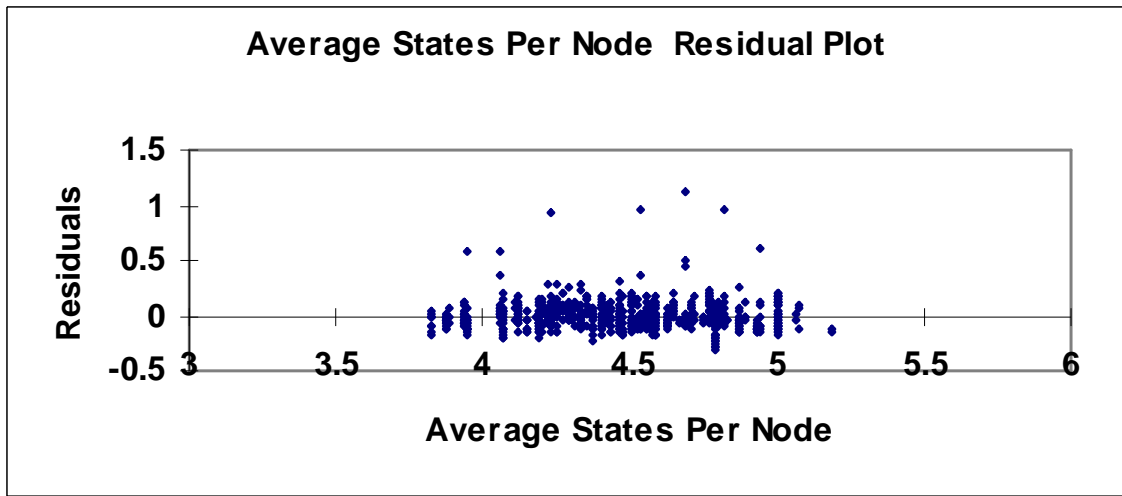


Figure 17: Particle Filter Average Speed Exponential Regression Model 4 Residuals versus the Average States per Node Graph

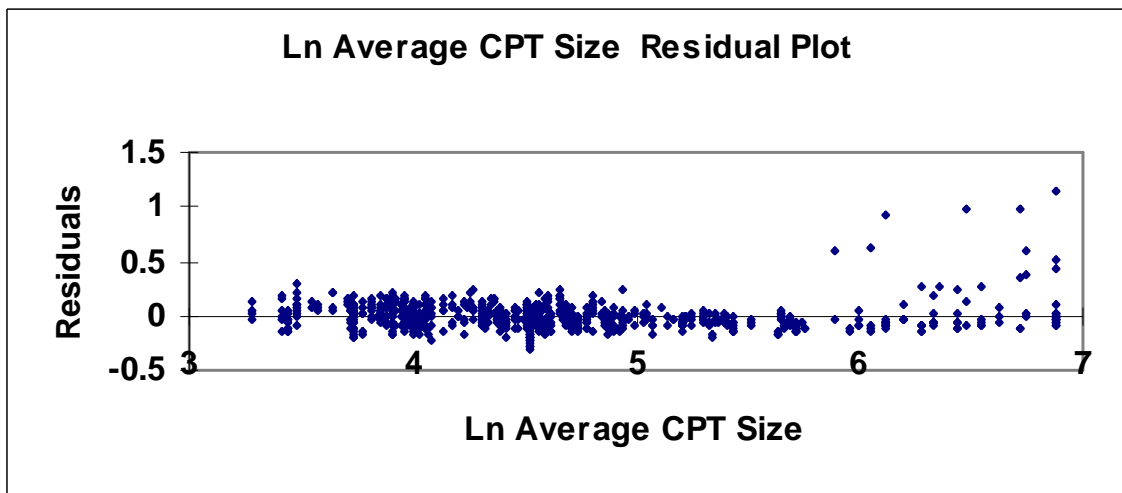


Figure 18: Particle Filter Average Speed Exponential Regression Model 4 Residuals versus the Natural Logarithm of the Average CPT Size Graph

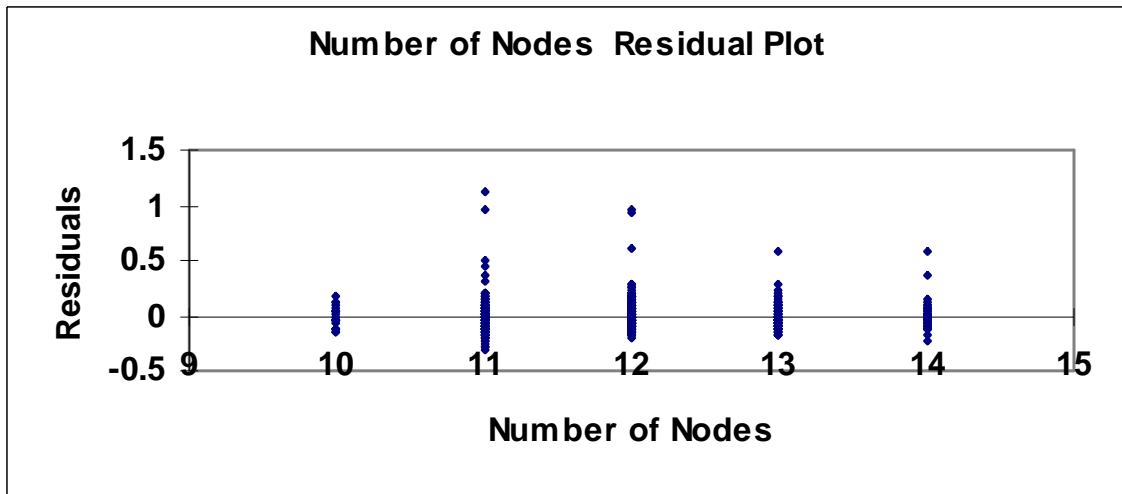


Figure 19: Particle Filter Average Speed Exponential Regression Model 4 Residuals versus Number of Nodes Graph

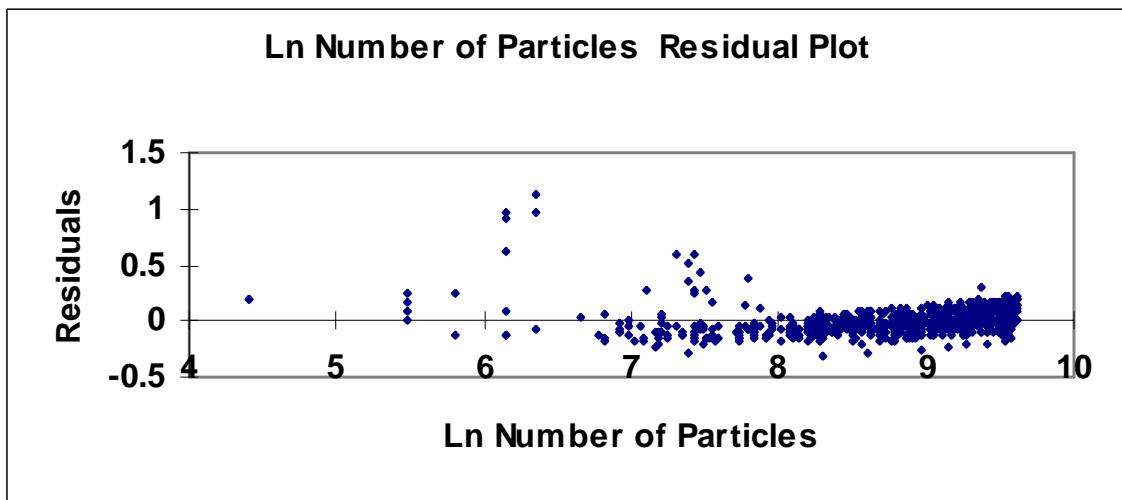


Figure 20: Particle Filter Average Speed Exponential Regression Model 4 Residuals versus the Natural Logarithm of the Number of Particles Graph

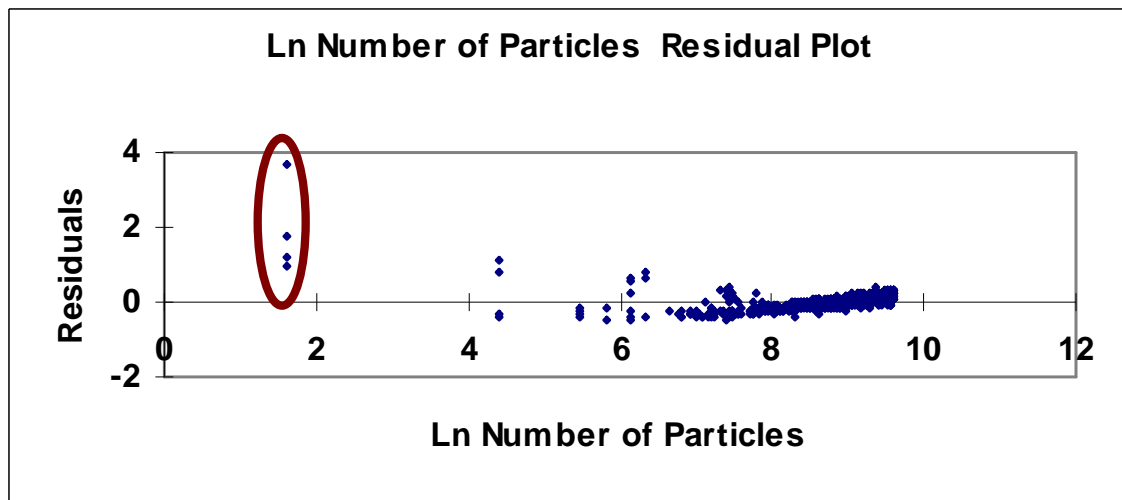


Figure 21: Particle Filter Average Speed Exponential Regression Model 4 Residuals versus the Natural Logarithm of the Number of Particles Graph Before Removal of Outliers

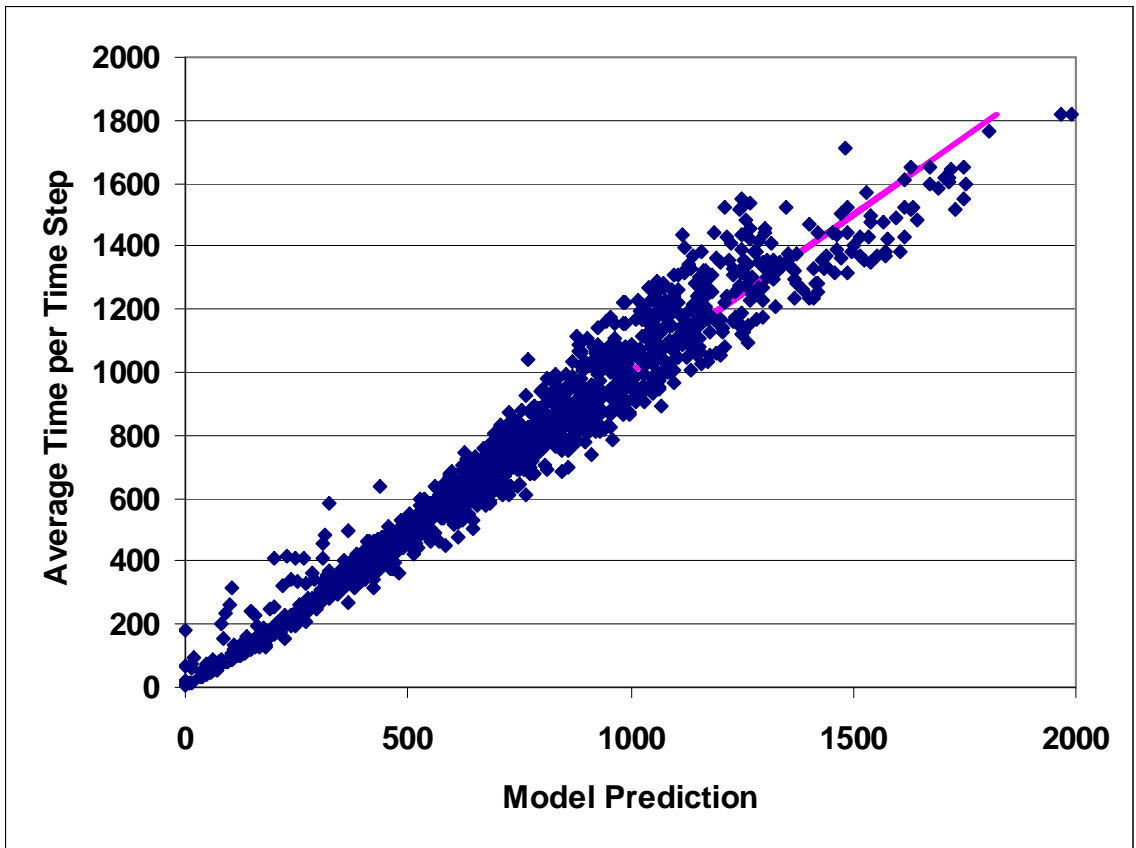


Figure 22: Particle Filter Average Speed Exponential Regression Model 4 Predicted versus Actual Graph

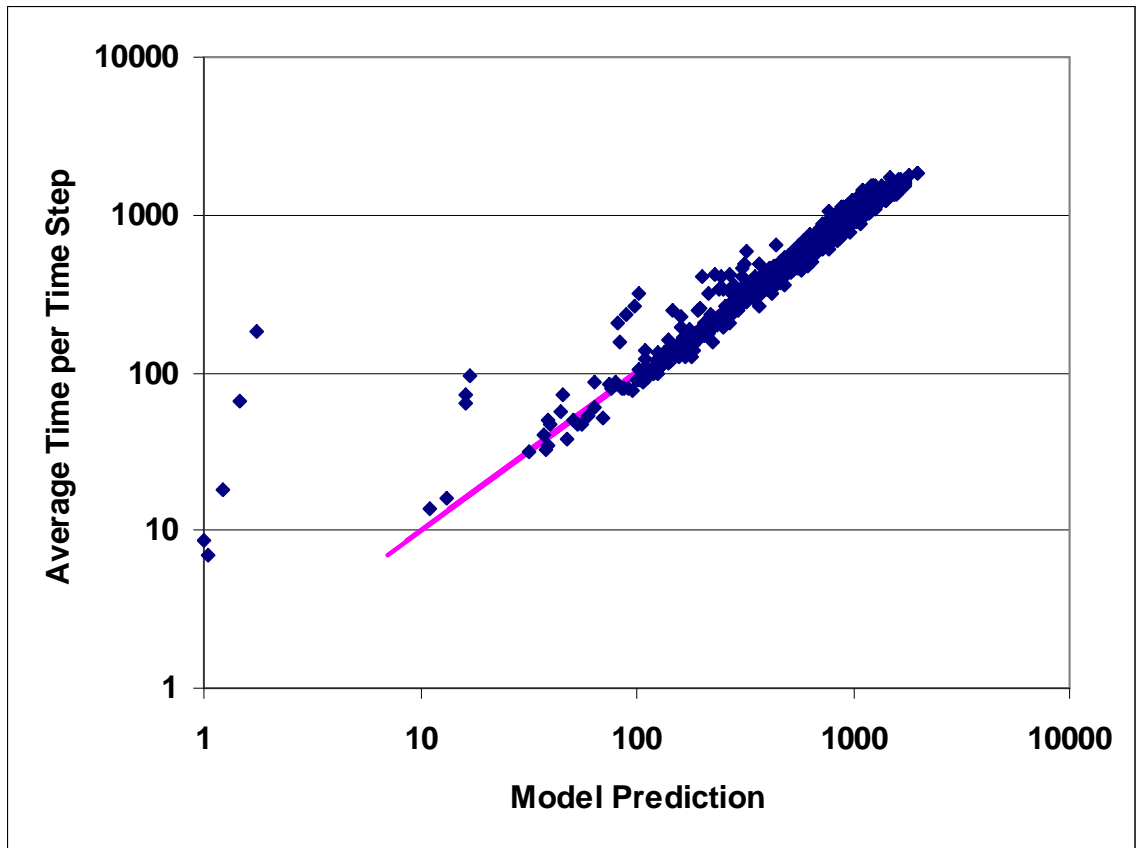


Figure 23: Particle Filter Average Speed Exponential Regression Model 4 Predicted versus Actual Graph in Logarithm Space

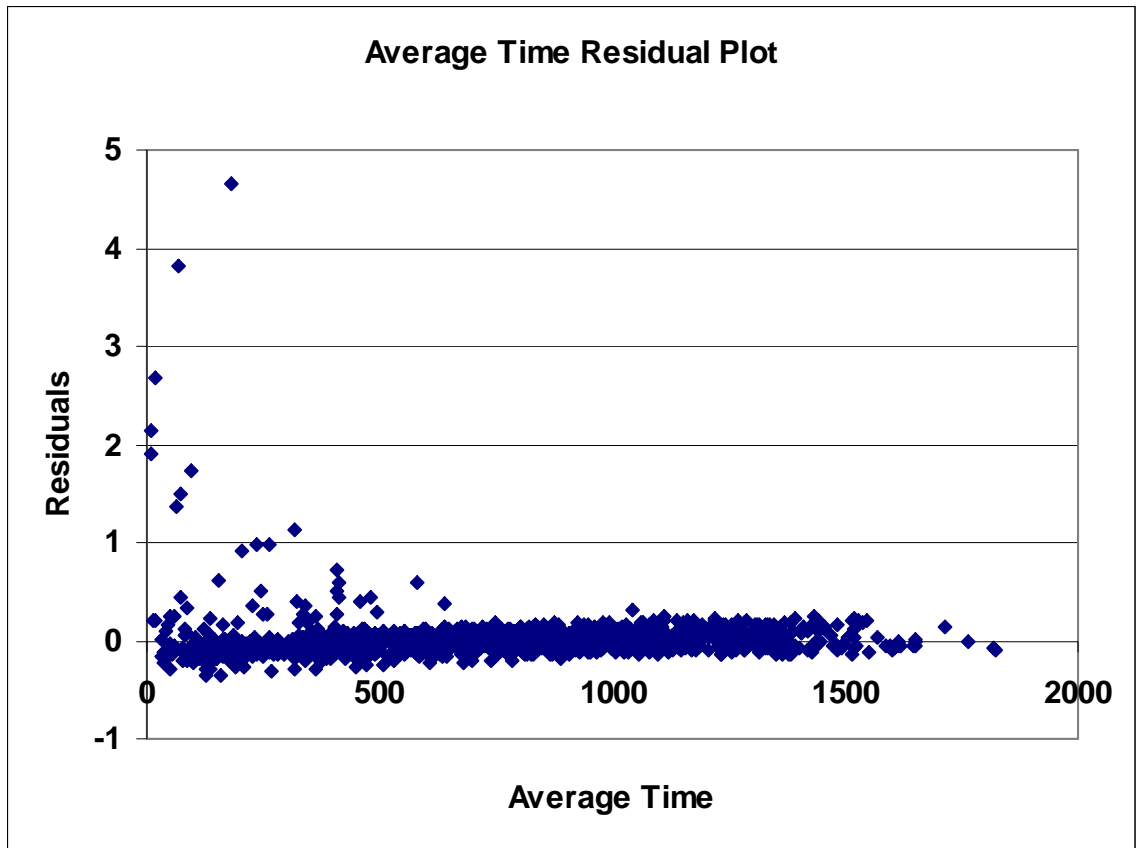


Figure 24: Particle Filter Average Speed Exponential Regression Model 4 Residuals Plot

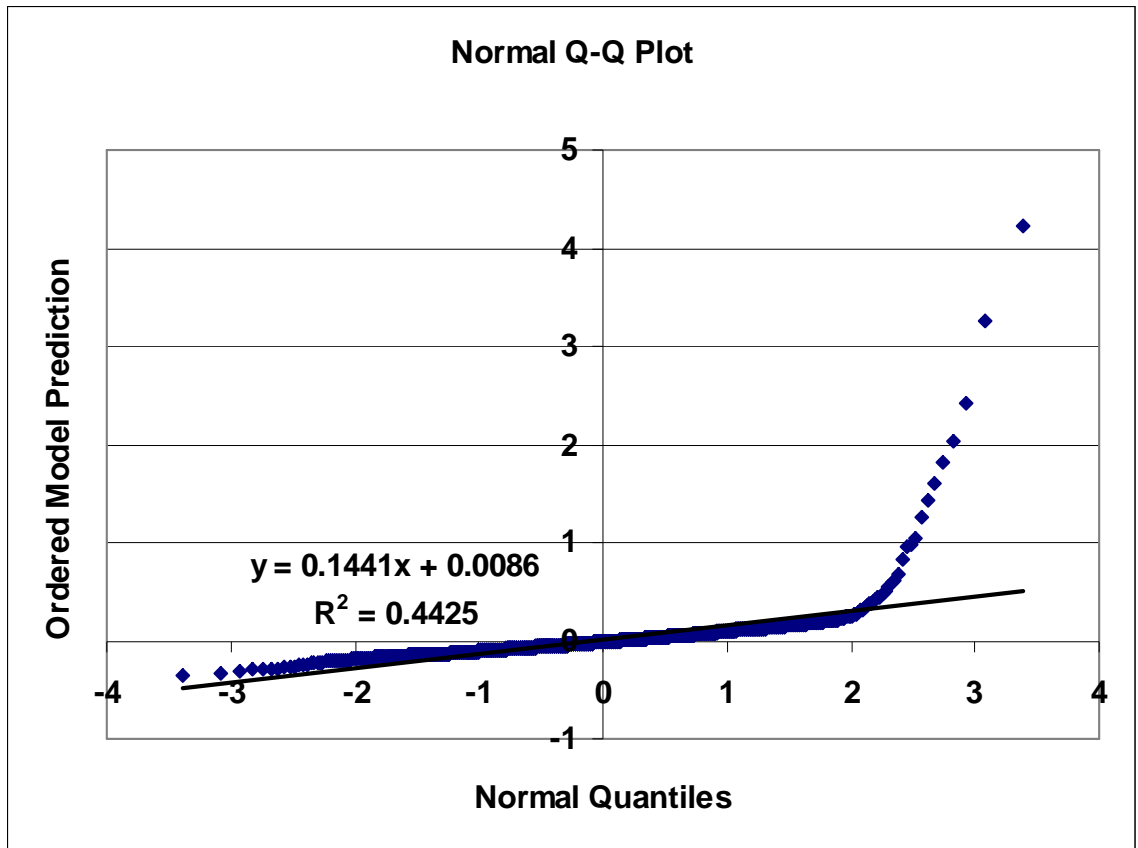


Figure 25: Particle Filter Average Speed Exponential Regression Model 4 Q-Q Plot

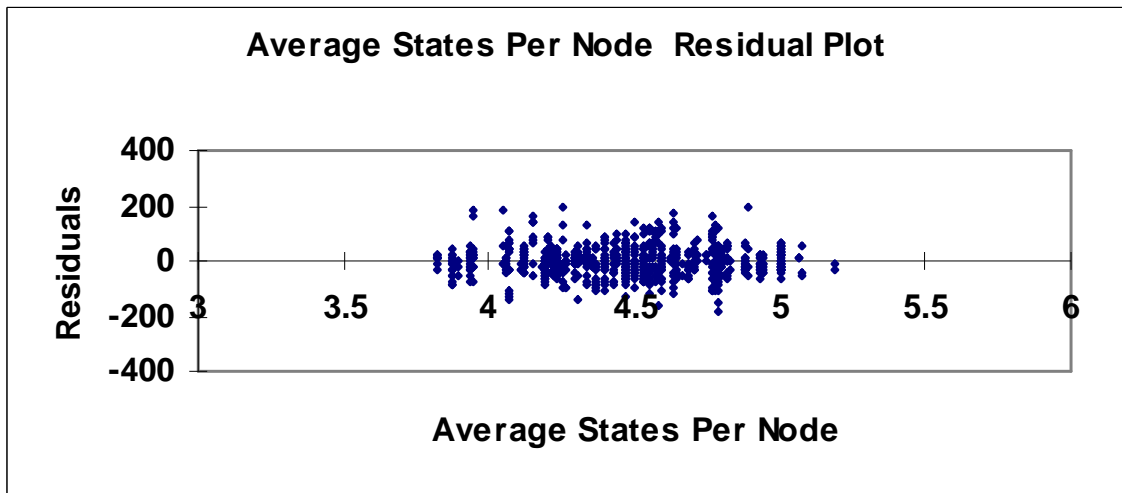


Figure 26: Particle Filter Average Speed Linear Regression Model 1 Residuals versus Average States per Node Graph

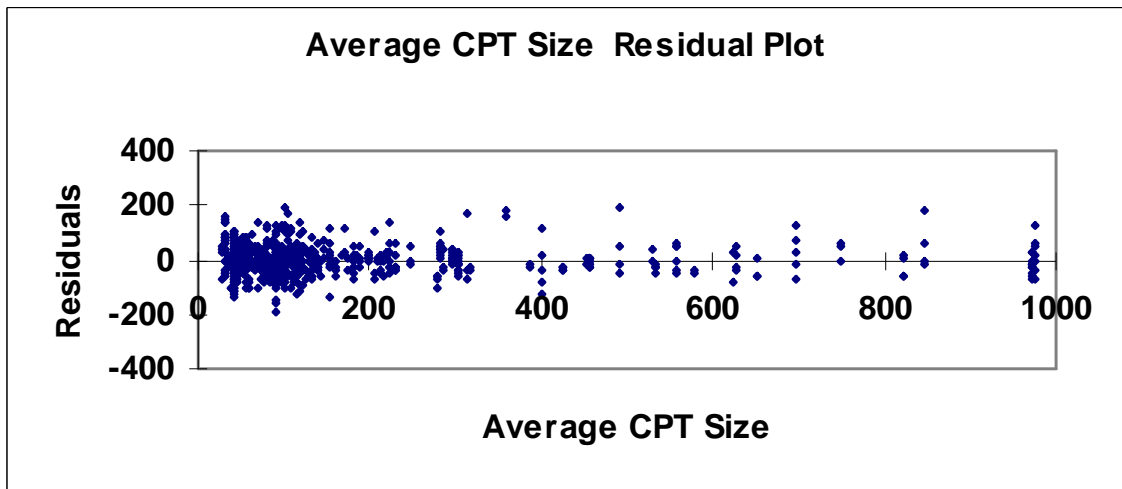


Figure 27: Particle Filter Average Speed Linear Regression Model 1 Residuals versus Average CPT Size Graph

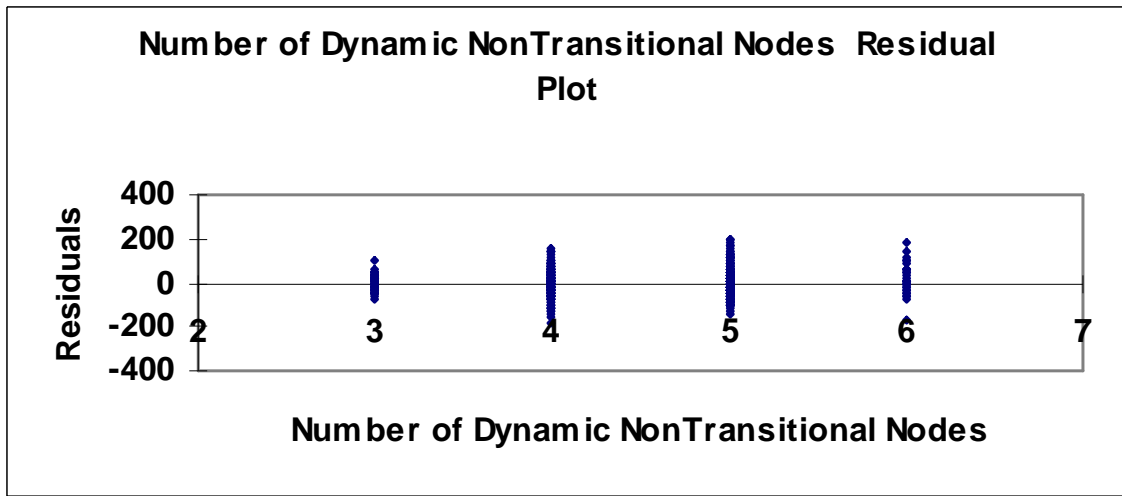


Figure 28: Particle Filter Average Speed Linear Regression Model 1 Residuals versus Number of Dynamic Non-Transitional Nodes Graph

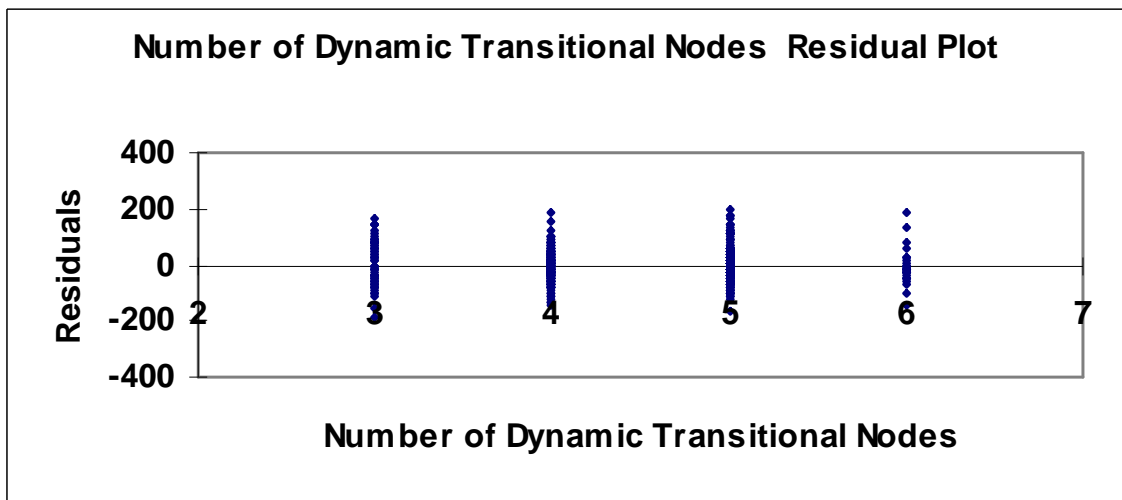


Figure 29: Particle Filter Average Speed Linear Regression Model 1 Residuals versus Number of Dynamic Transitional Nodes Graph

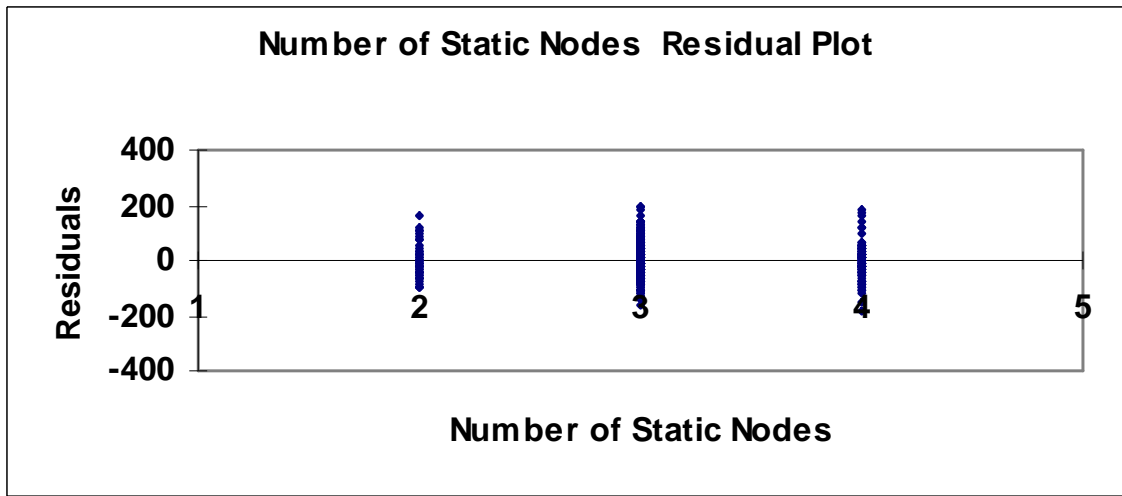


Figure 30: Particle Filter Average Speed Linear Regression Model 1 Residuals versus Number of Static Nodes Graph

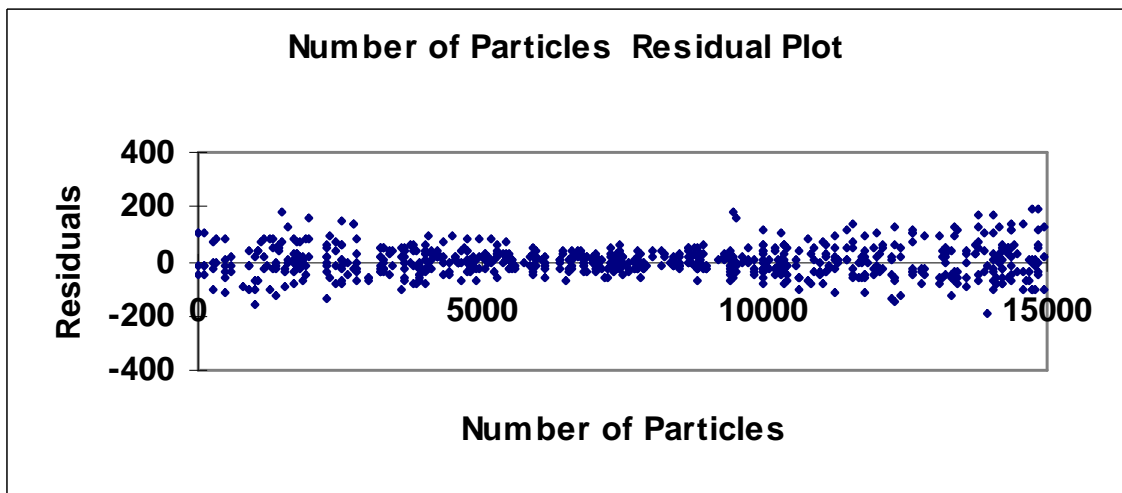


Figure 31: Particle Filter Average Speed Linear Regression Model 1 Residuals versus Number of Particles Graph

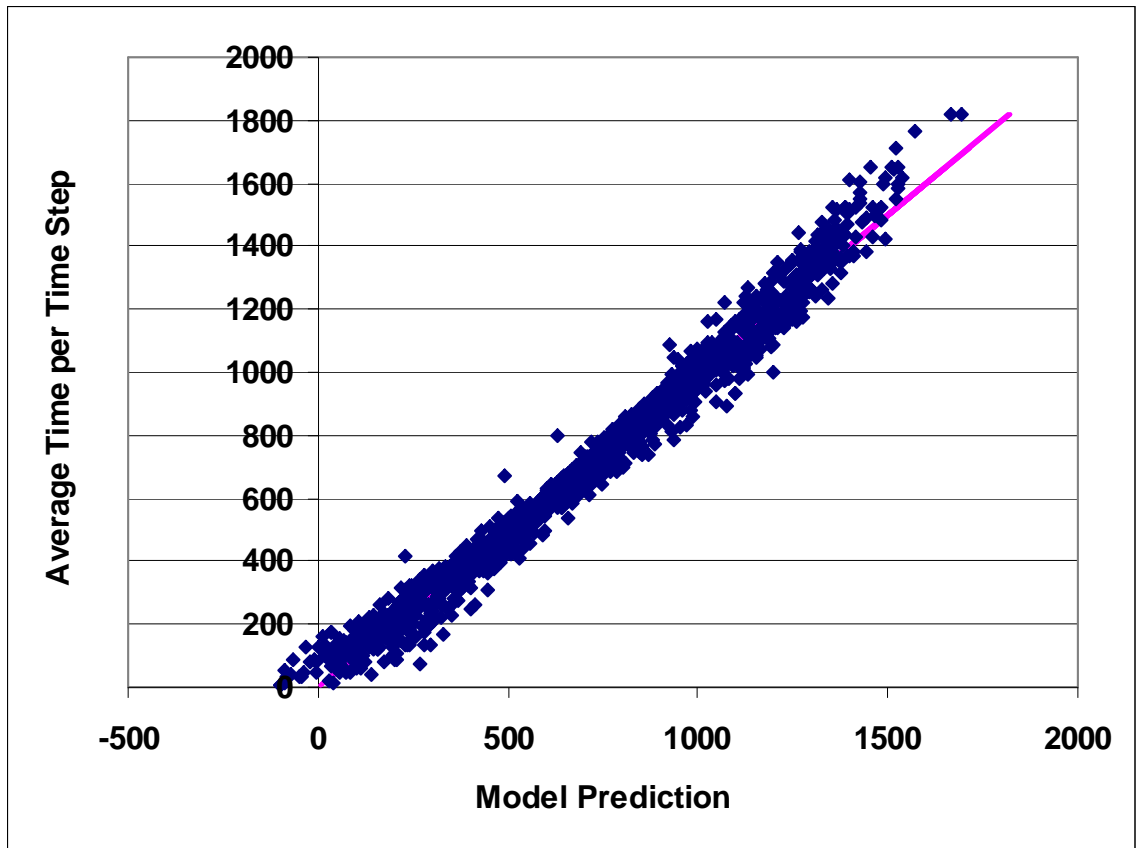


Figure 32: Particle Filter Average Speed Linear Regression Model 1 Predicted versus Actual Graph

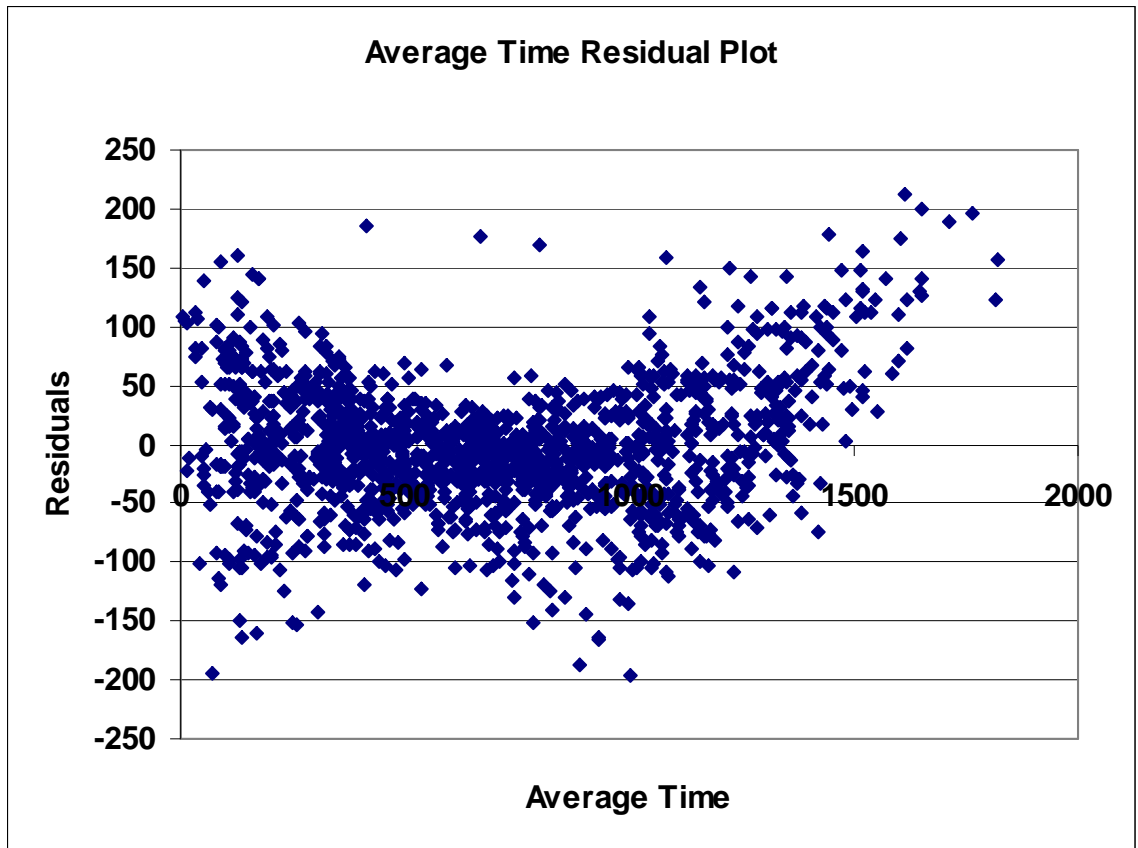


Figure 33: Particle Filter Average Speed Linear Regression Model 1 Residuals Plot

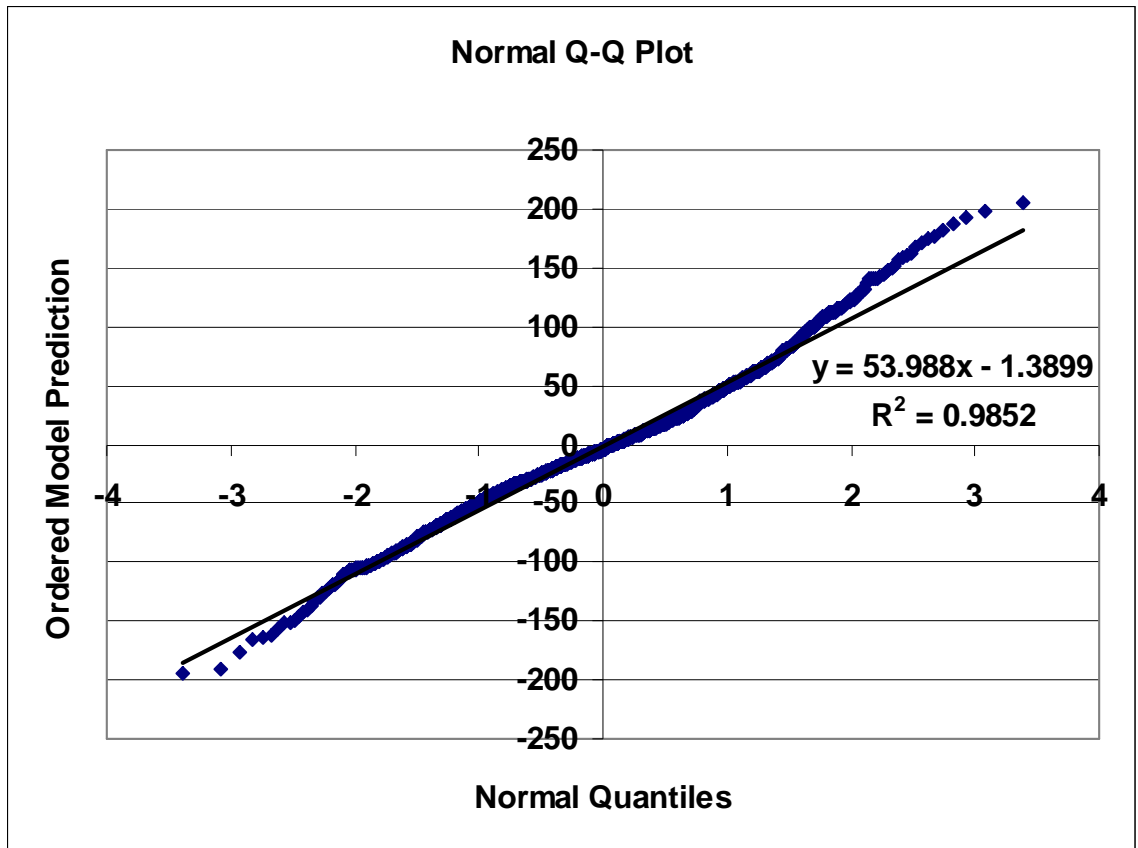


Figure 34: Particle Filter Average Speed Linear Regression Model 1 Q-Q Plot

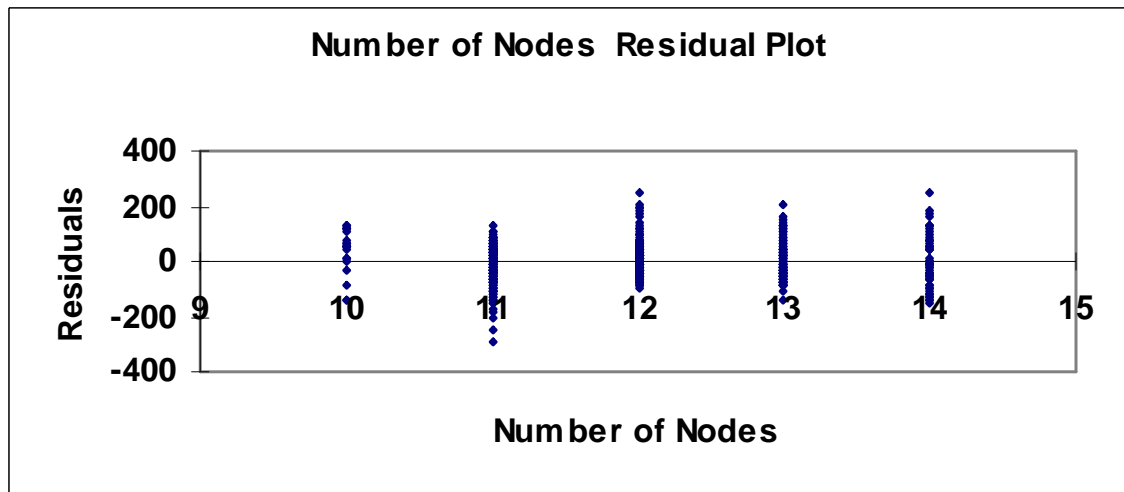


Figure 35: Particle Filter Average Speed Linear Regression Model 2 Residuals versus Number of Nodes Graph

Appendix B: Regression Diagnostics for the Particle Filter Variance of Speed of
Inference Models

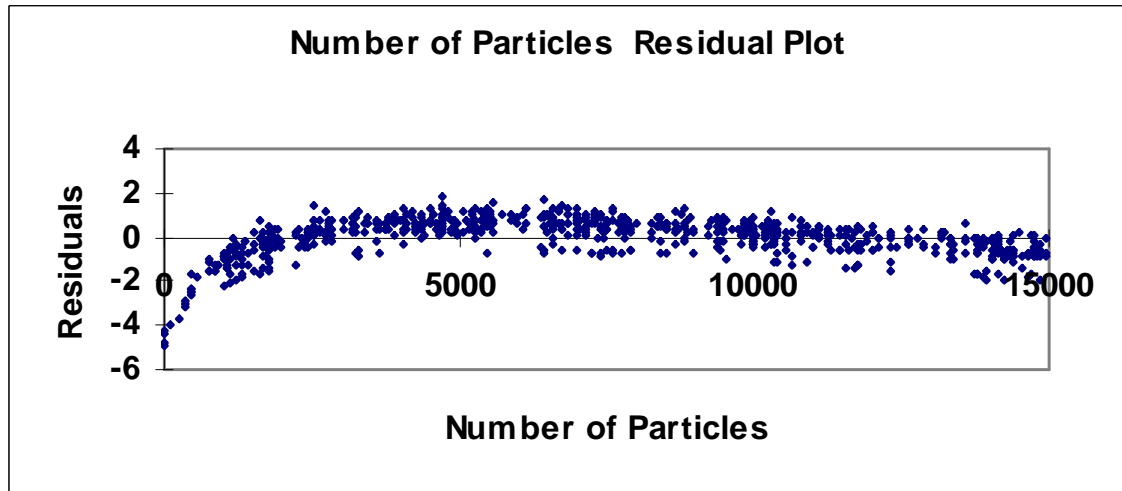


Figure 36: Particle Filter Variance of Speed Exponential Regression Model 1 Residuals versus Number of Particles Graph

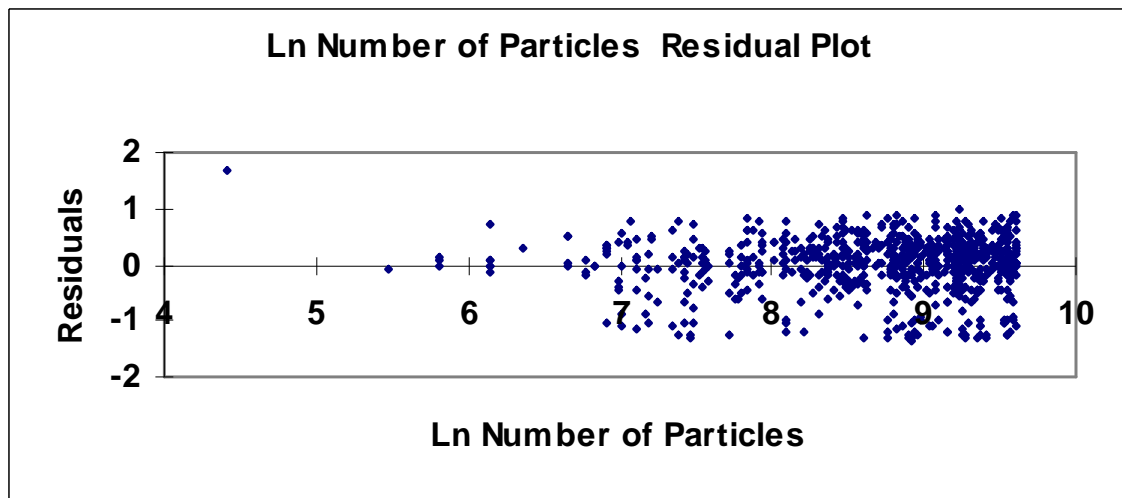


Figure 37: Particle Filter Variance of Speed Exponential Regression Model 2 Residuals versus Number of Particles Graph

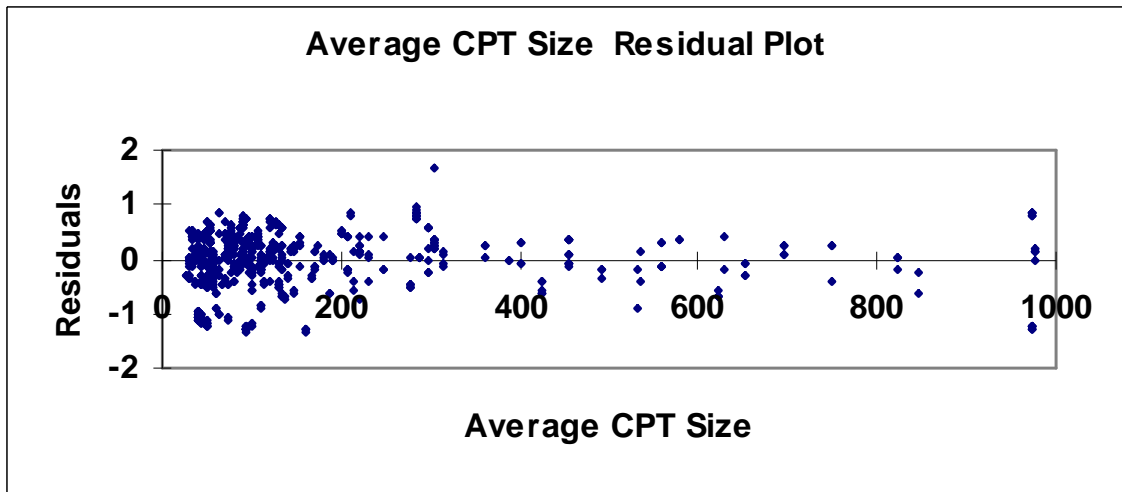


Figure 38: Particle Filter Variance of Speed Exponential Regression Model 2 Residuals versus Average CPT Size Graph

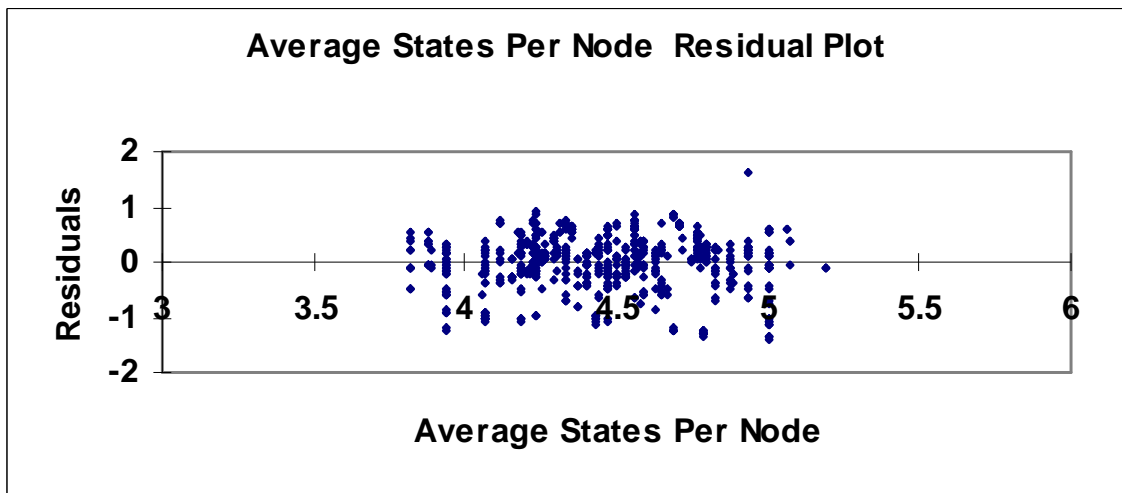


Figure 39: Particle Filter Variance of Speed Exponential Regression Model 3 Residuals versus Average States per Node Graph

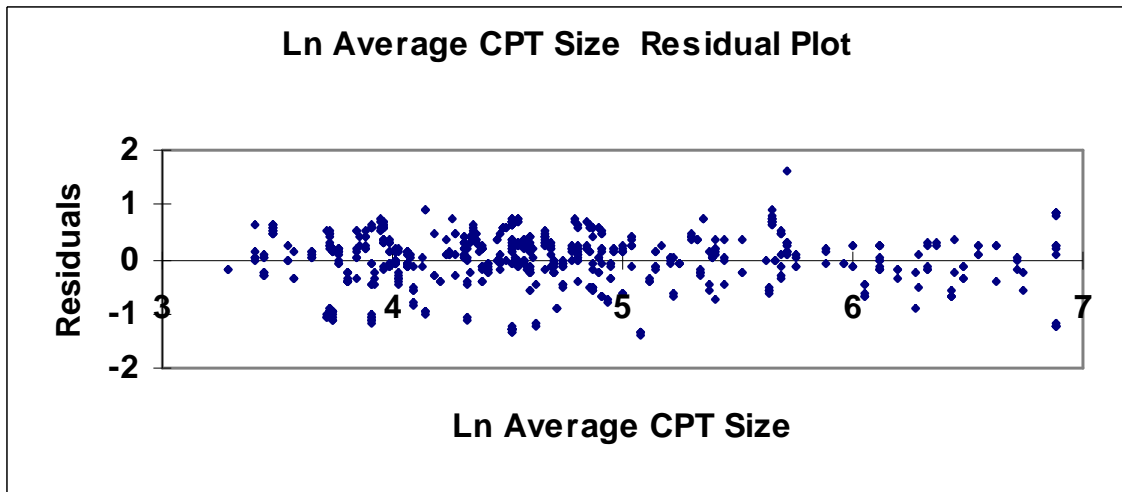


Figure 40: Particle Filter Variance of Speed Exponential Regression Model 3 Residuals versus the Natural Logarithm of the Average CPT Size Graph

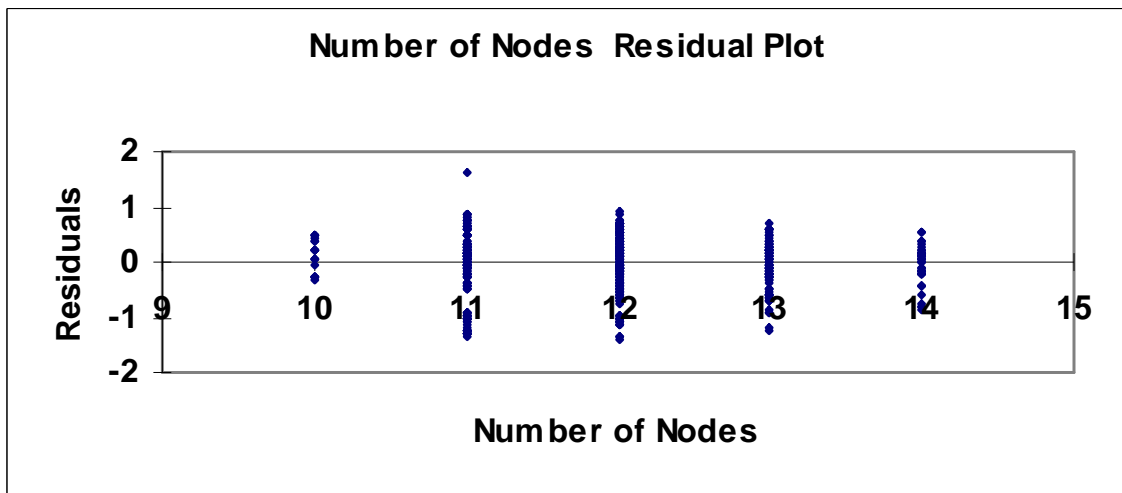


Figure 41: Particle Filter Variance of Speed Exponential Regression Model 3 Residuals versus Number of Nodes Graph

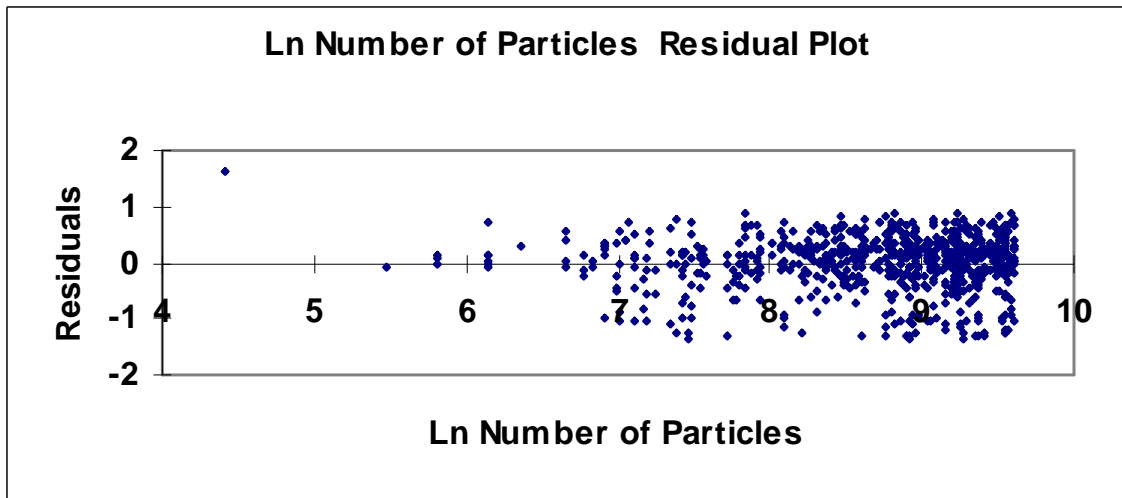


Figure 42: Particle Filter Variance of Speed Exponential Regression Model 3 Residuals versus the Natural Logarithm of the Number of Particles Graph

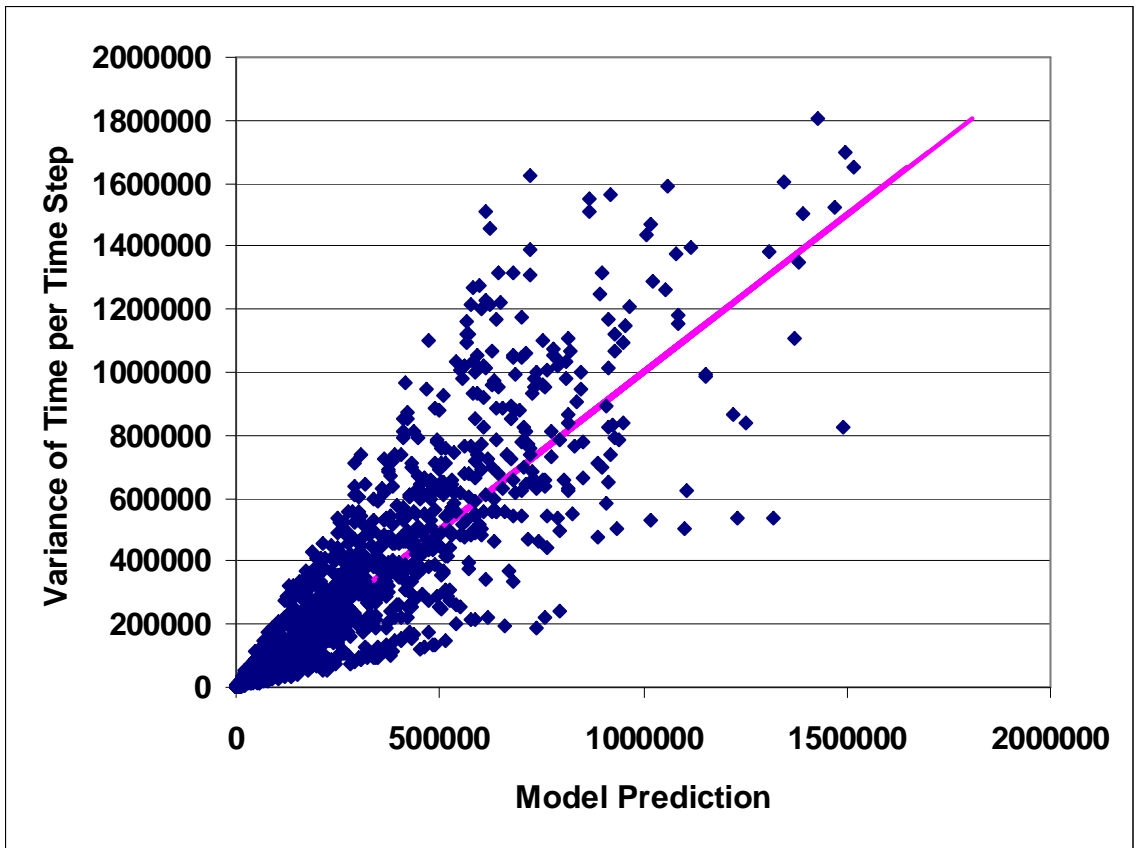


Figure 43: Particle Filter Variance of Speed Exponential Regression Model 3 Predicted versus Actual Graph

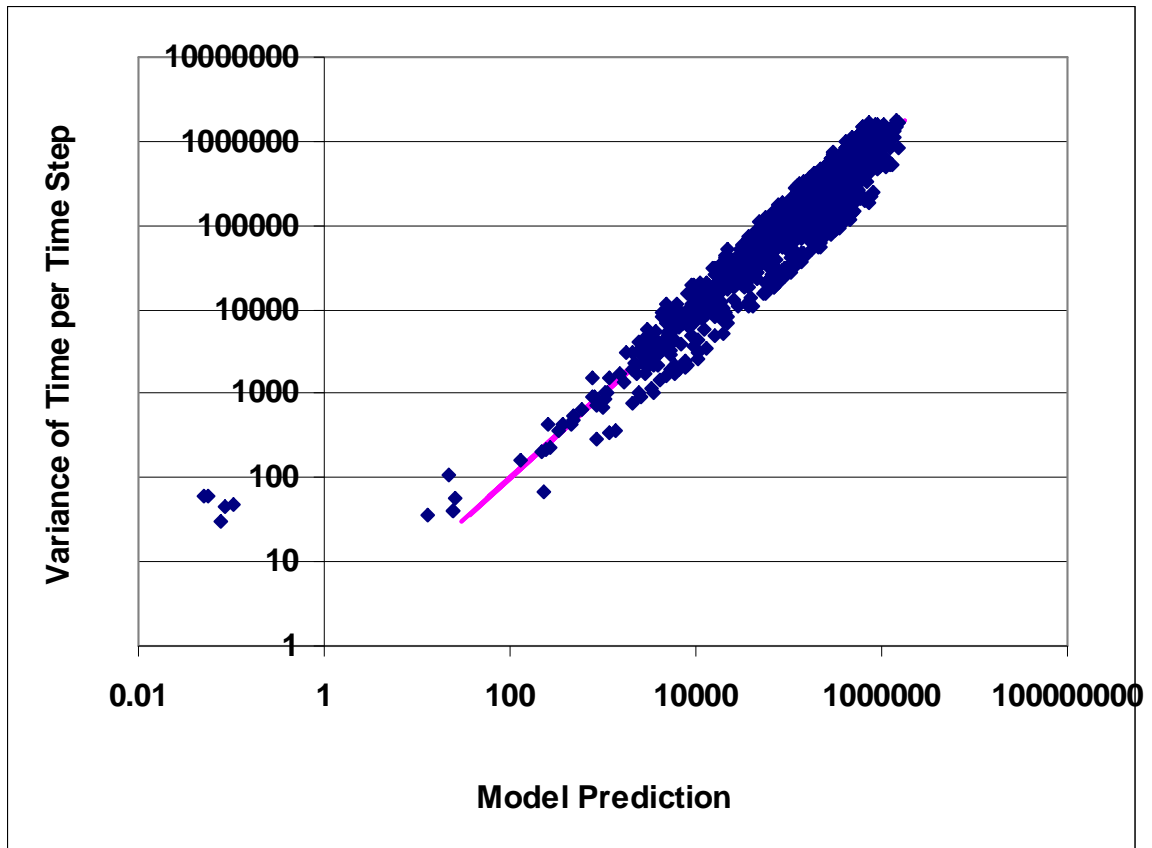


Figure 44: Particle Filter Variance of the Speed Exponential Regression Model 3 Predicted versus Actual Graph in Logarithm Space

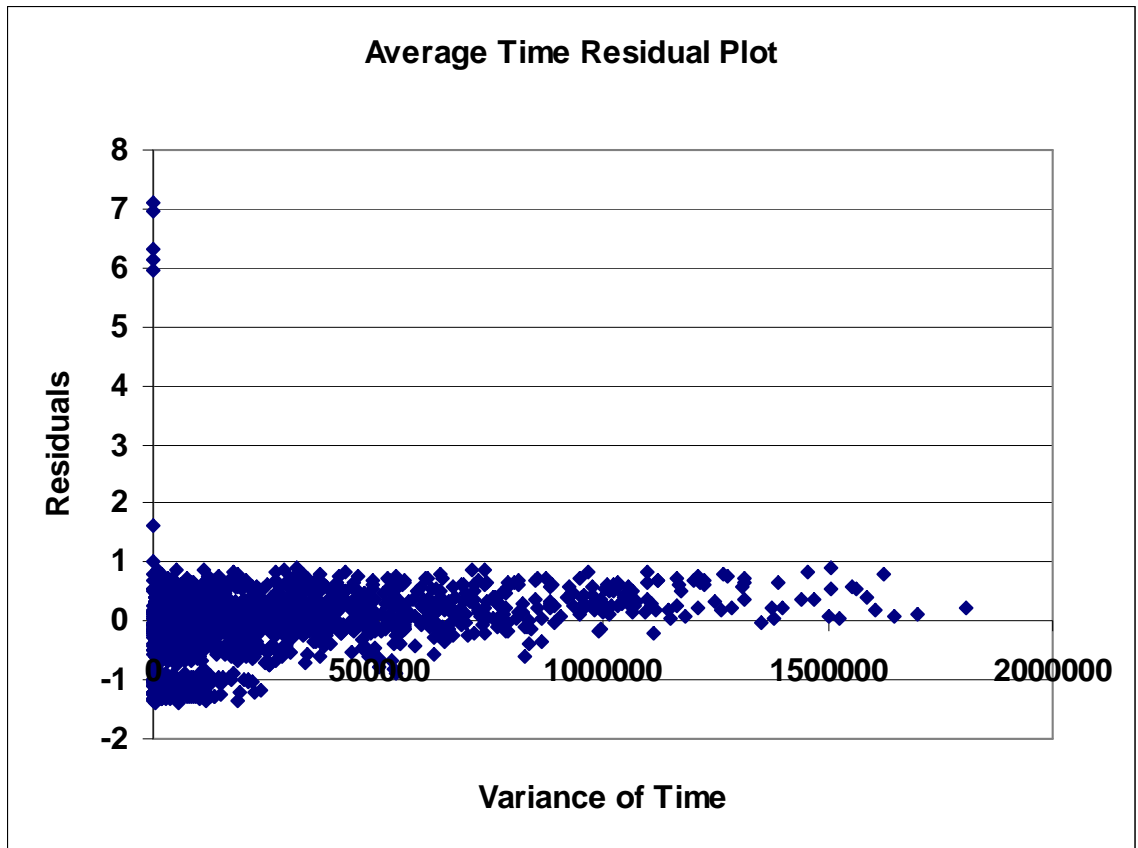


Figure 45: Particle Filter Variance of Speed Exponential Regression Model 3 Residuals Plot

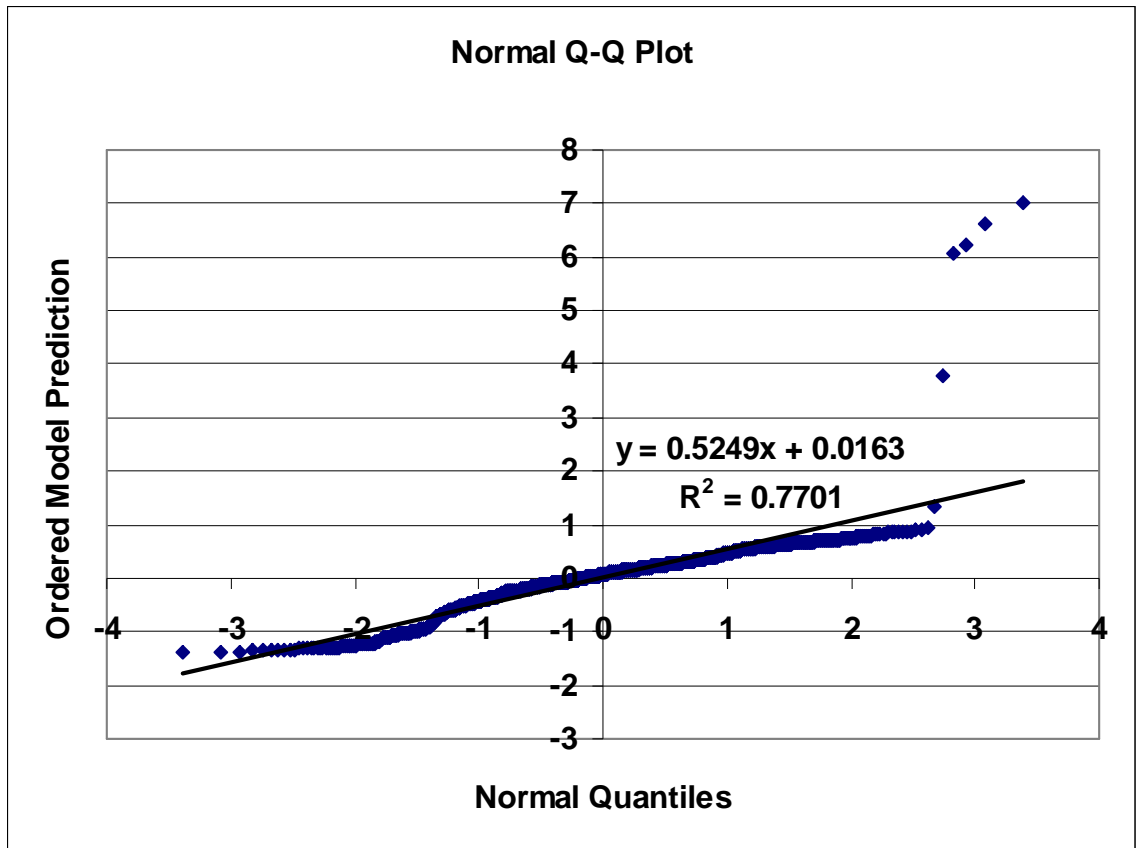


Figure 46: Particle Filter Variance of Speed Exponential Regression Model 3 Q-Q Plot

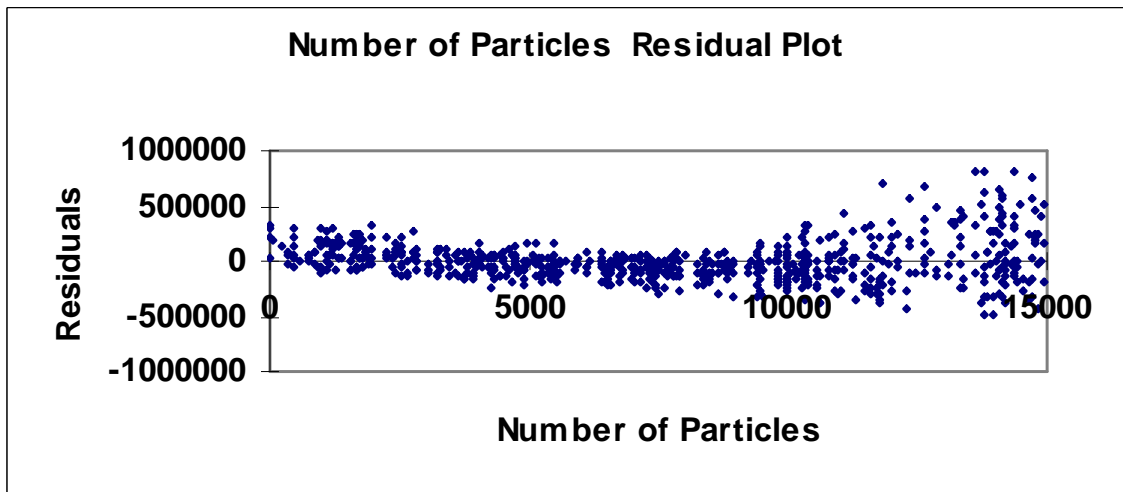


Figure 47: Particle Filter Variance of Speed Linear Regression Model 1 Residuals versus Number of Particles Graph

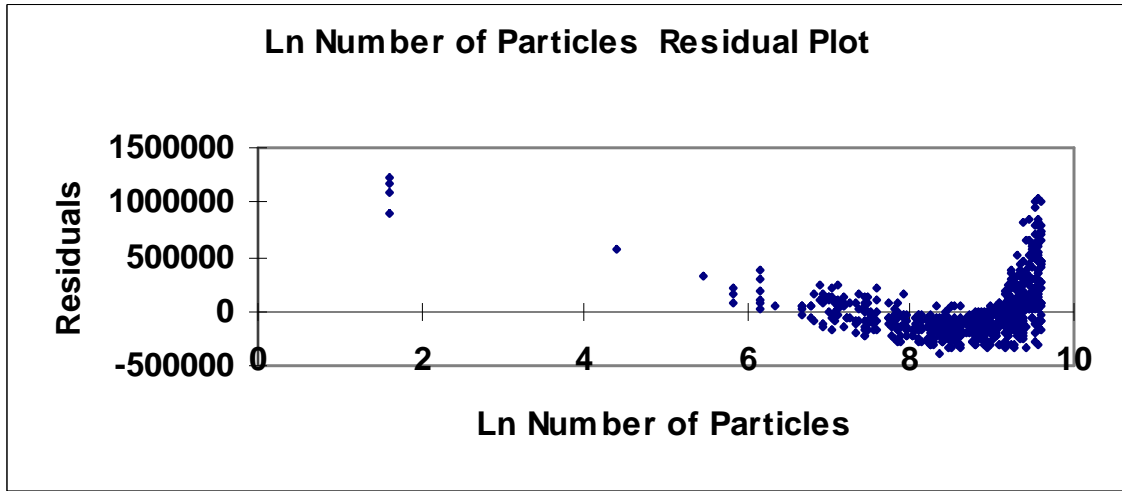


Figure 48: Particle Filter Variance of Speed Linear Regression Model 2 Residuals versus Number of Particles Graph

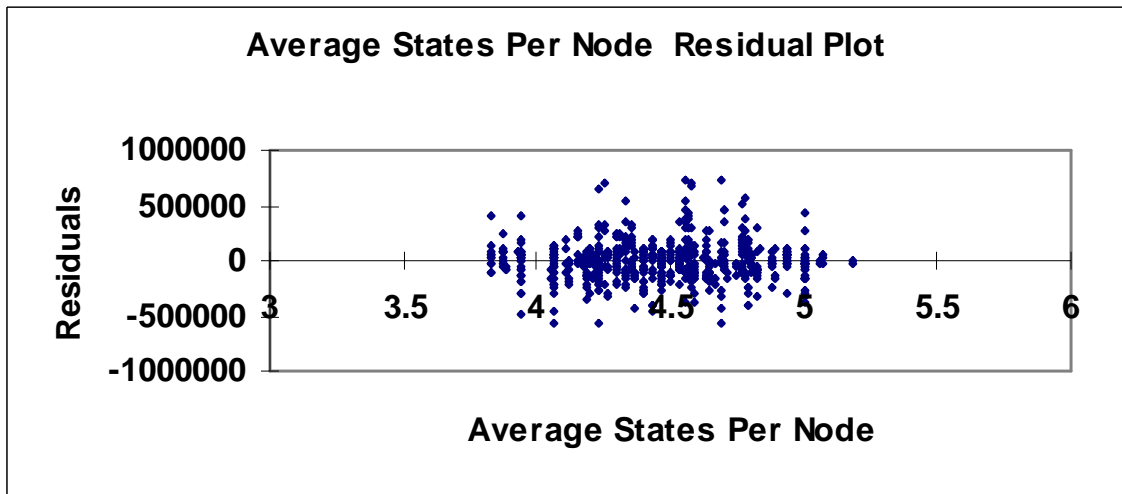


Figure 49: Particle Filter Variance of Speed Linear Regression Model 3 Residuals versus Average States per Node Graph

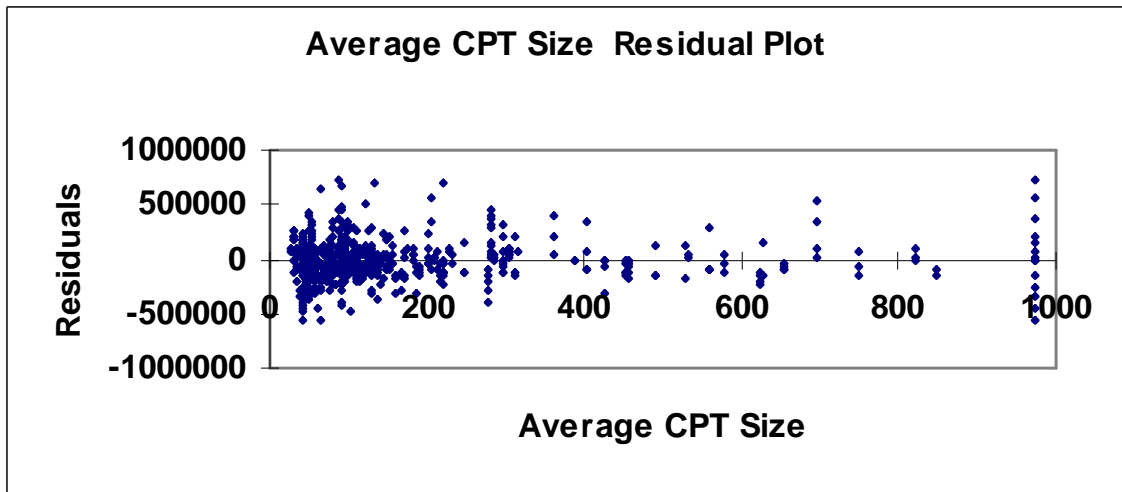


Figure 50: Particle Filter Variance of Speed Linear Regression Model 3 Residuals versus Average CPT Size Graph

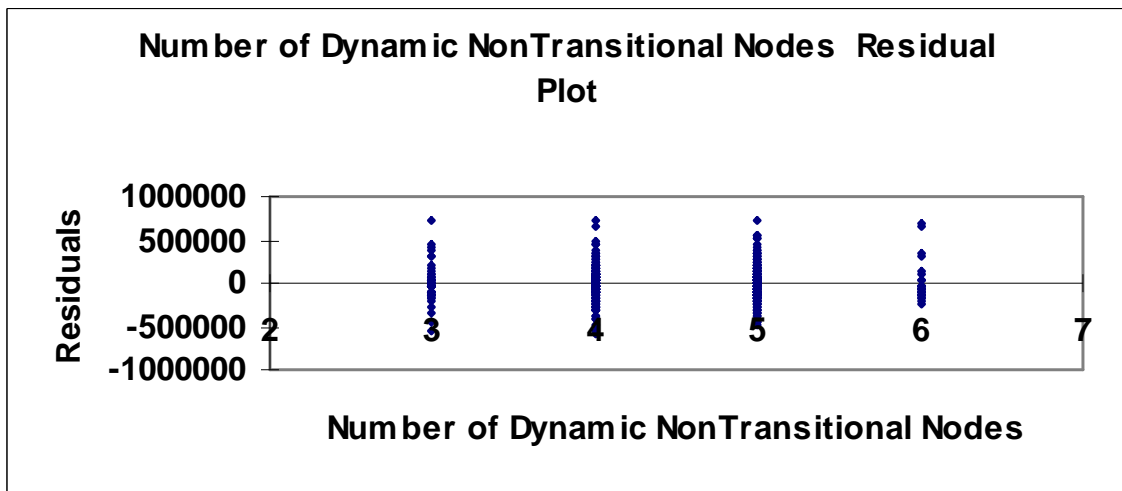


Figure 51: Particle Filter Variance of Speed Linear Regression Model 3 Residuals versus Number of Dynamic Non-Transitional Nodes Graph

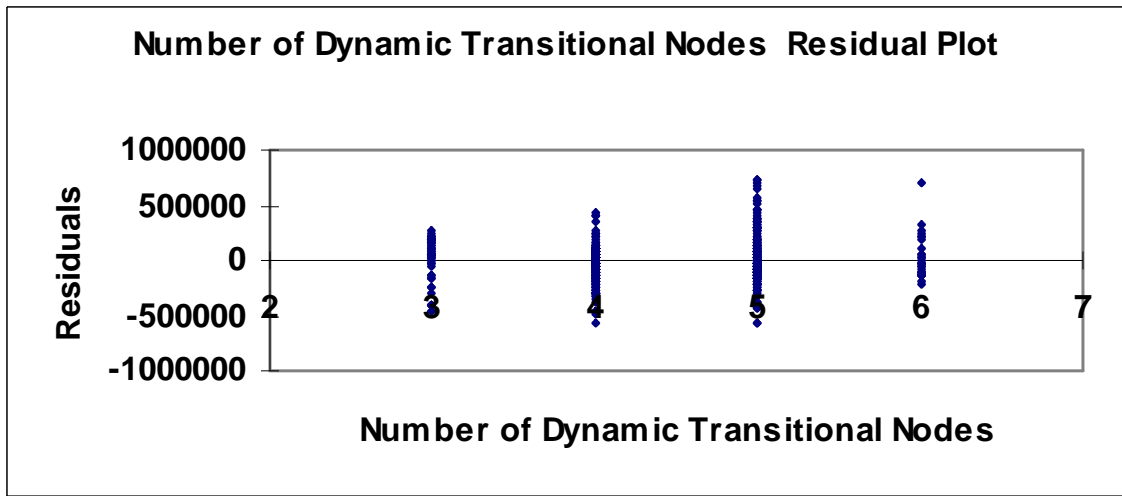


Figure 52: Particle Filter Variance of Speed Linear Regression Model 3 Residuals versus Number of Dynamic Transitional Nodes Graph

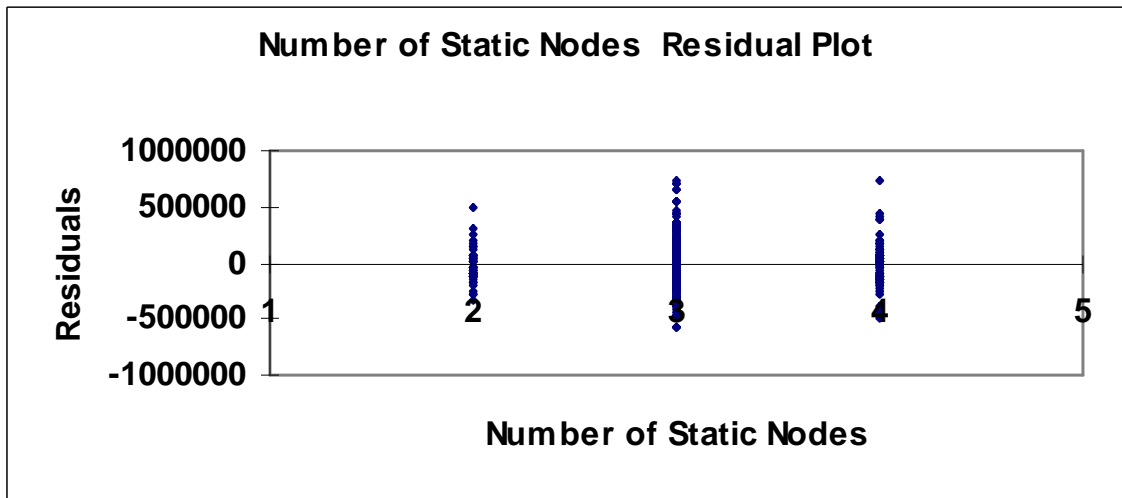


Figure 53: Particle Filter Variance of Speed Linear Regression Model 3 Residuals versus Number of Static Nodes Graph

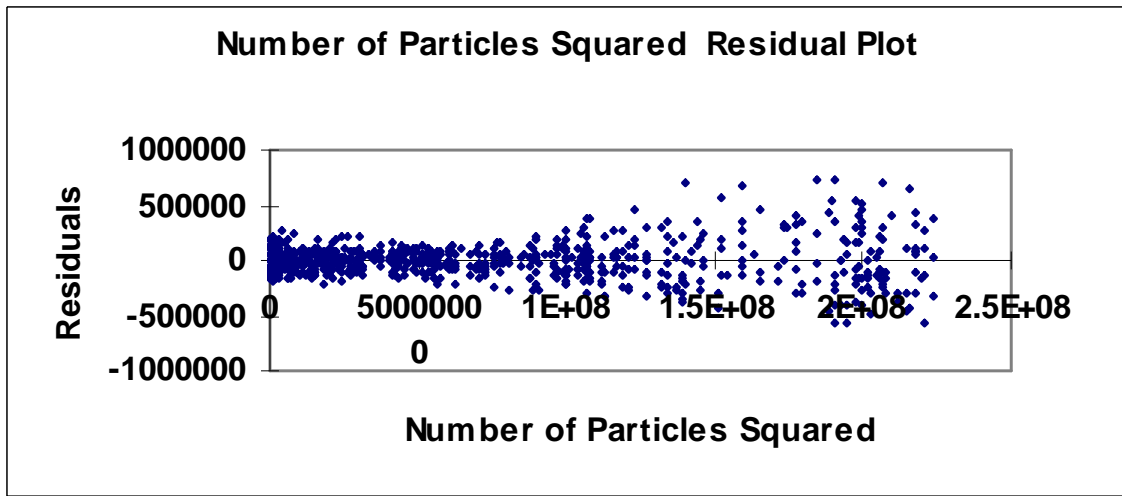


Figure 54: Particle Filter Variance of Speed Linear Regression Model 3 Residuals versus Number of Particles Squared Graph

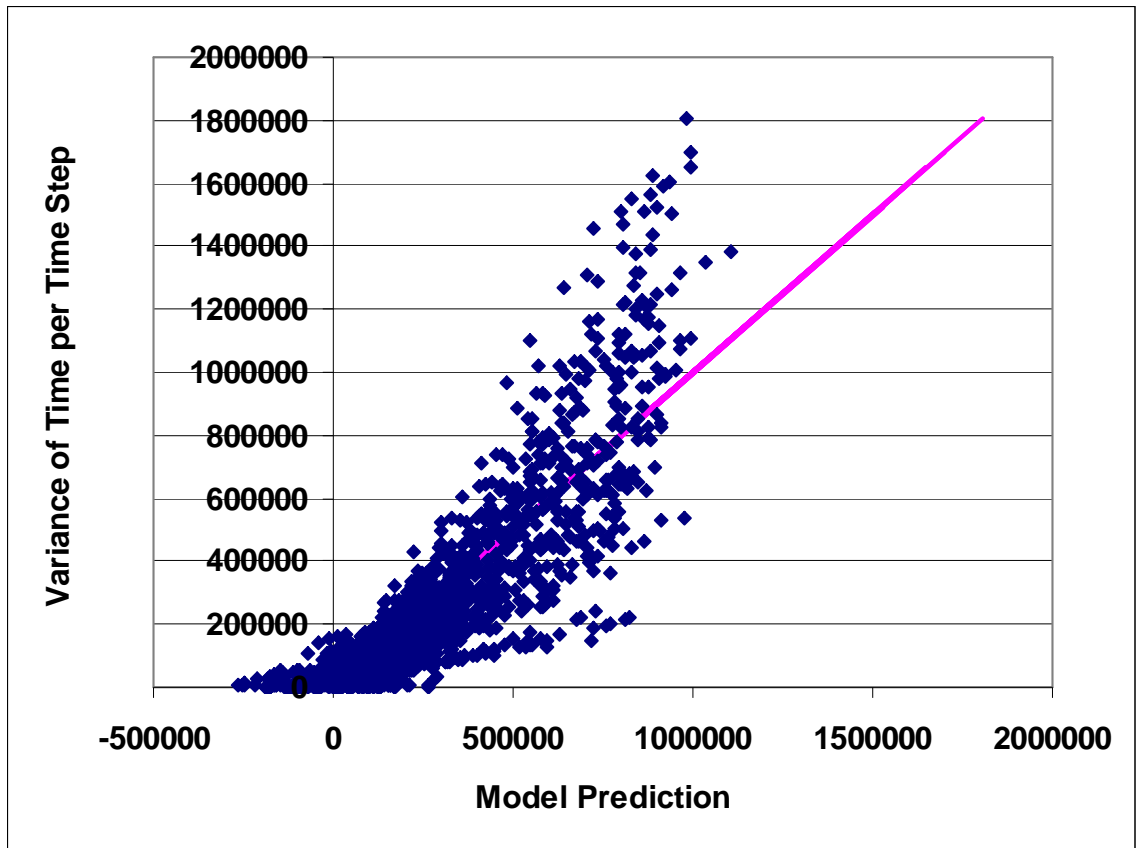


Figure 55: Particle Filter Variance of Speed Linear Regression Model 3 Predicted versus Actual Graph

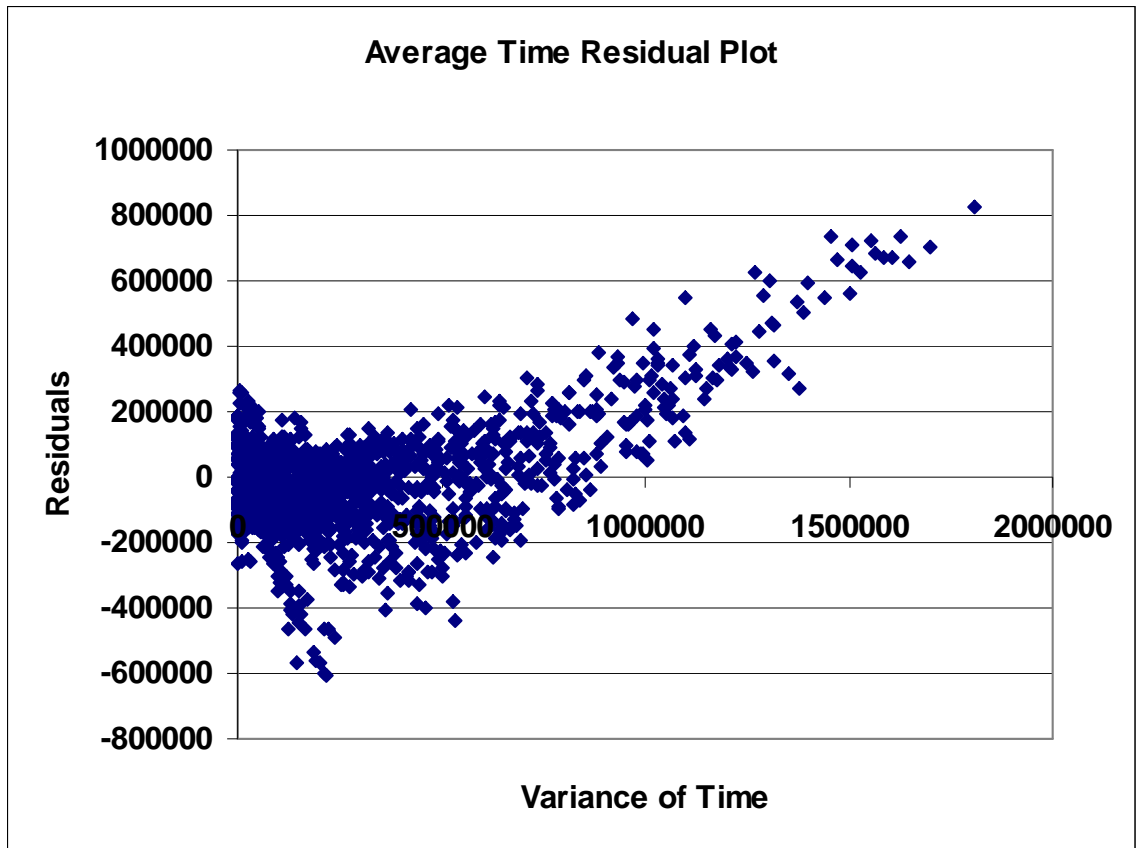


Figure 56: Particle Filter Variance of Speed Linear Regression Model 3 Residuals Plot

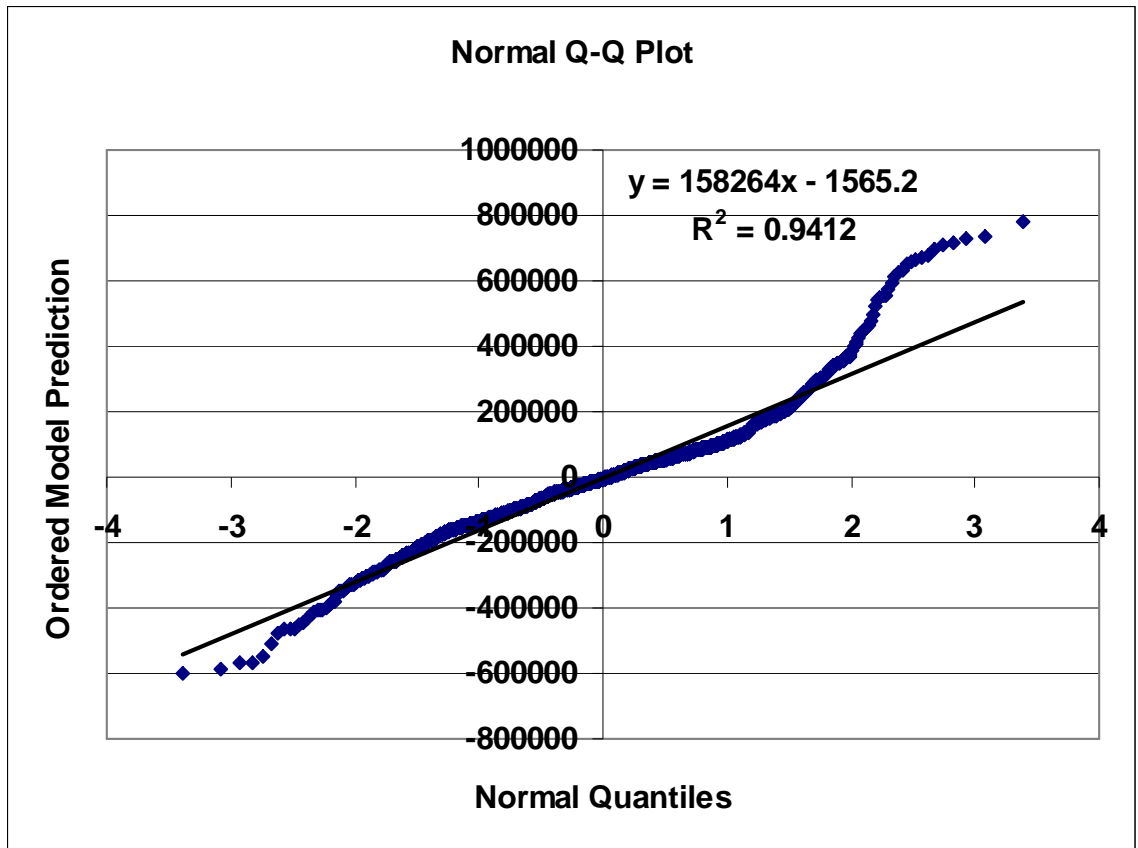


Figure 57: Particle Filter Variance of Speed Linear Regression Model 3 Q-Q Plot

Appendix C: Regression Diagnostics for the Boyen-Koller Speed of Inference Models

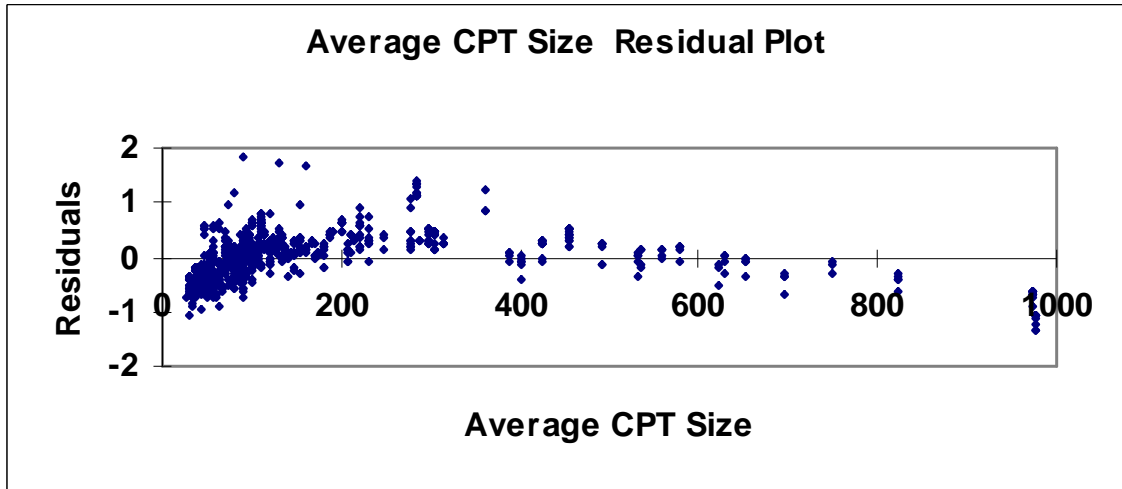


Figure 58: Boyen-Koller over SPI Average Speed Exponential Regression Model 1 Residuals versus Average CPT Size Graph

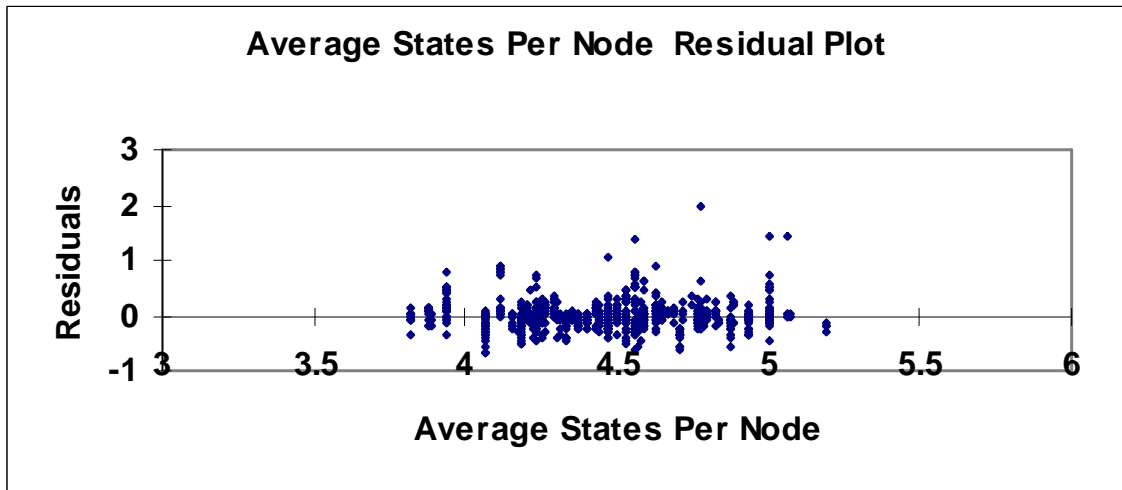


Figure 59: Boyen-Koller over SPI Average Speed Exponential Regression Model 2 Residuals versus States per Node Graph

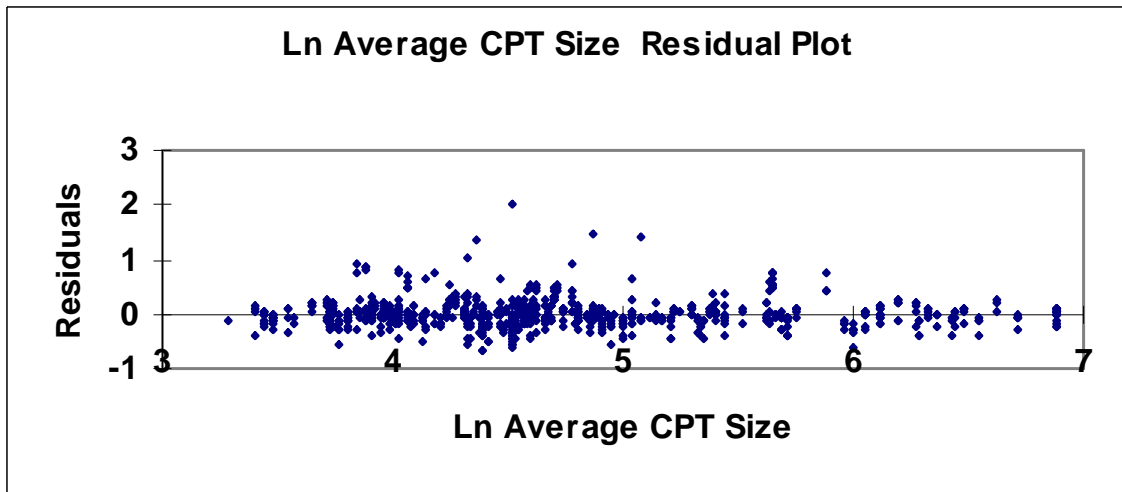


Figure 60: Boyen-Koller over SPI Average Speed Exponential Regression Model 2 Residuals versus Natural Logarithm of the Average CPT Size Graph

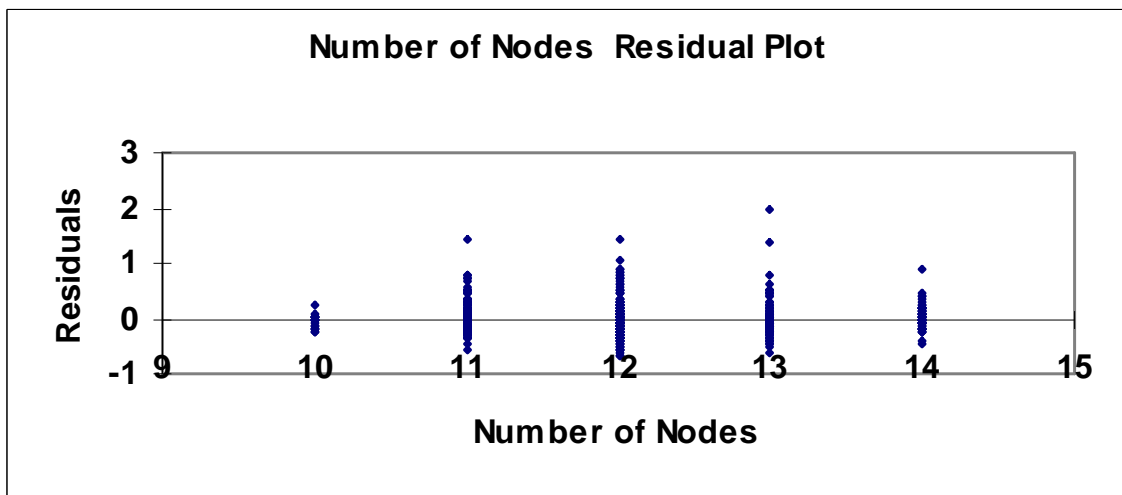


Figure 61: Boyen-Koller over SPI Average Speed Exponential Regression Model 2 Residuals versus Number of Nodes Graph

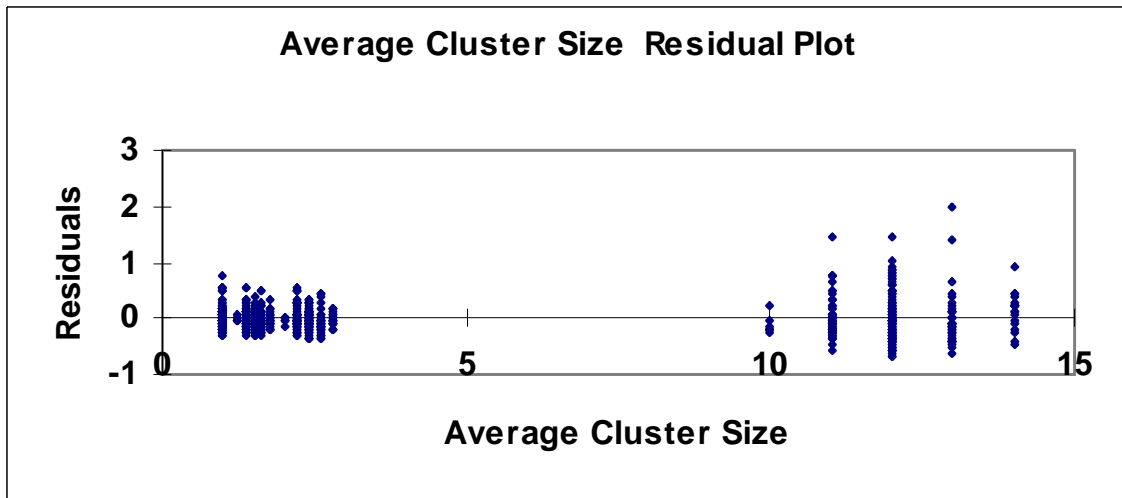


Figure 62: Boyen-Koller over SPI Average Speed Exponential Regression Model 2 Residuals versus Average Cluster Size Graph

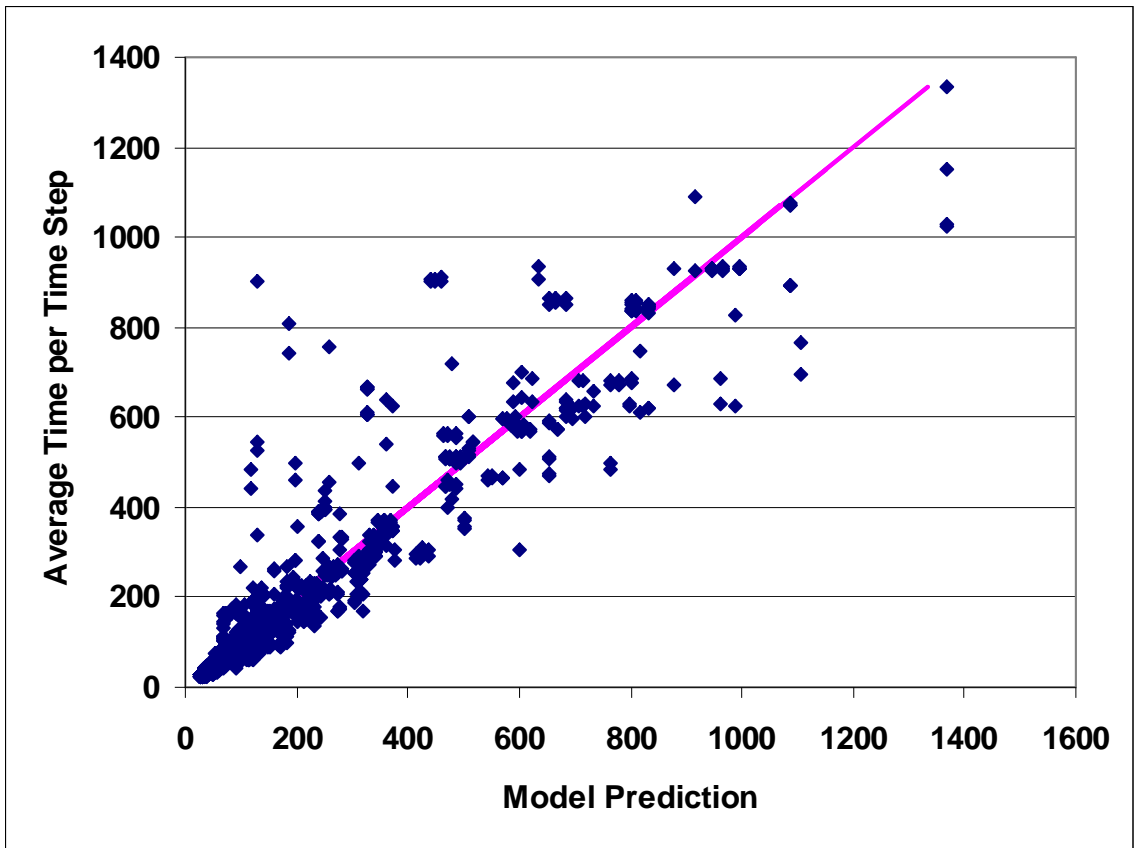


Figure 63: Boyen-Koller over SPI Average Speed Exponential Regression Model 2 Predicted versus Actual Graph

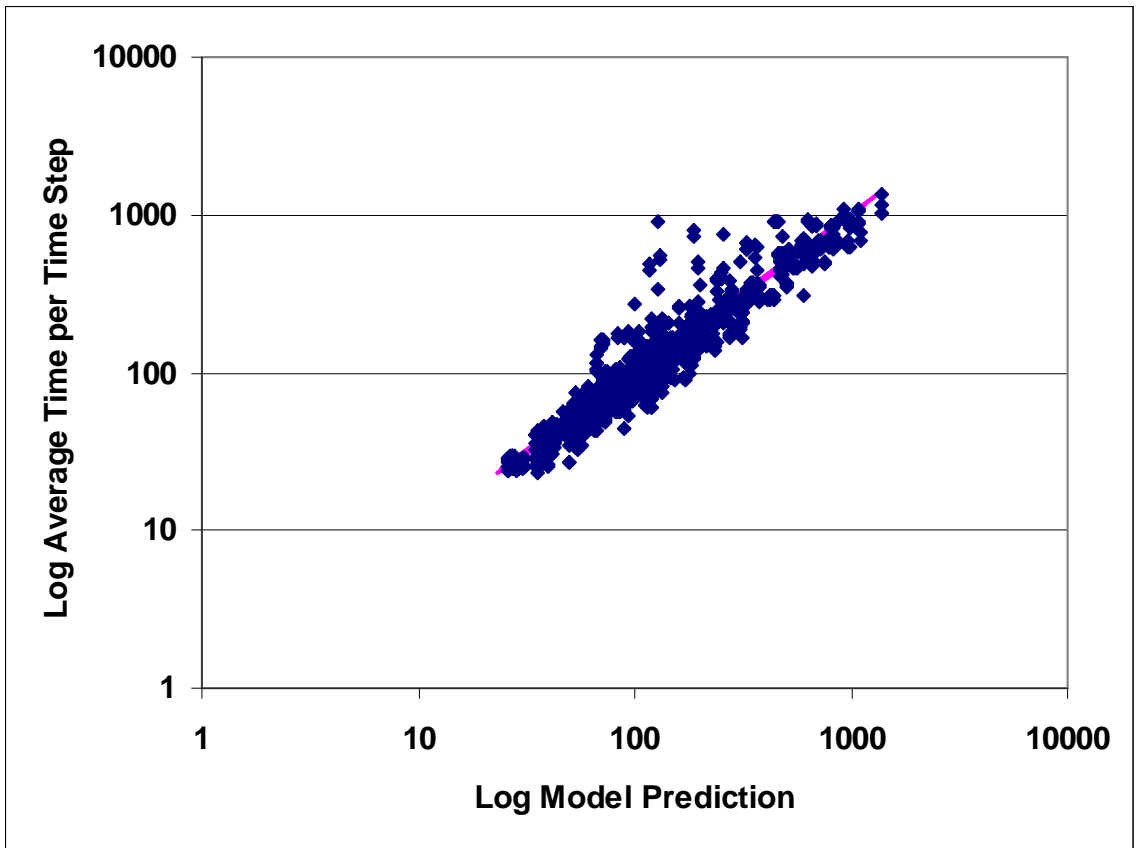


Figure 64: Boyen-Koller over SPI Average Speed Exponential Regression Model 2 Predicted versus Actual Graph in Logarithm Space

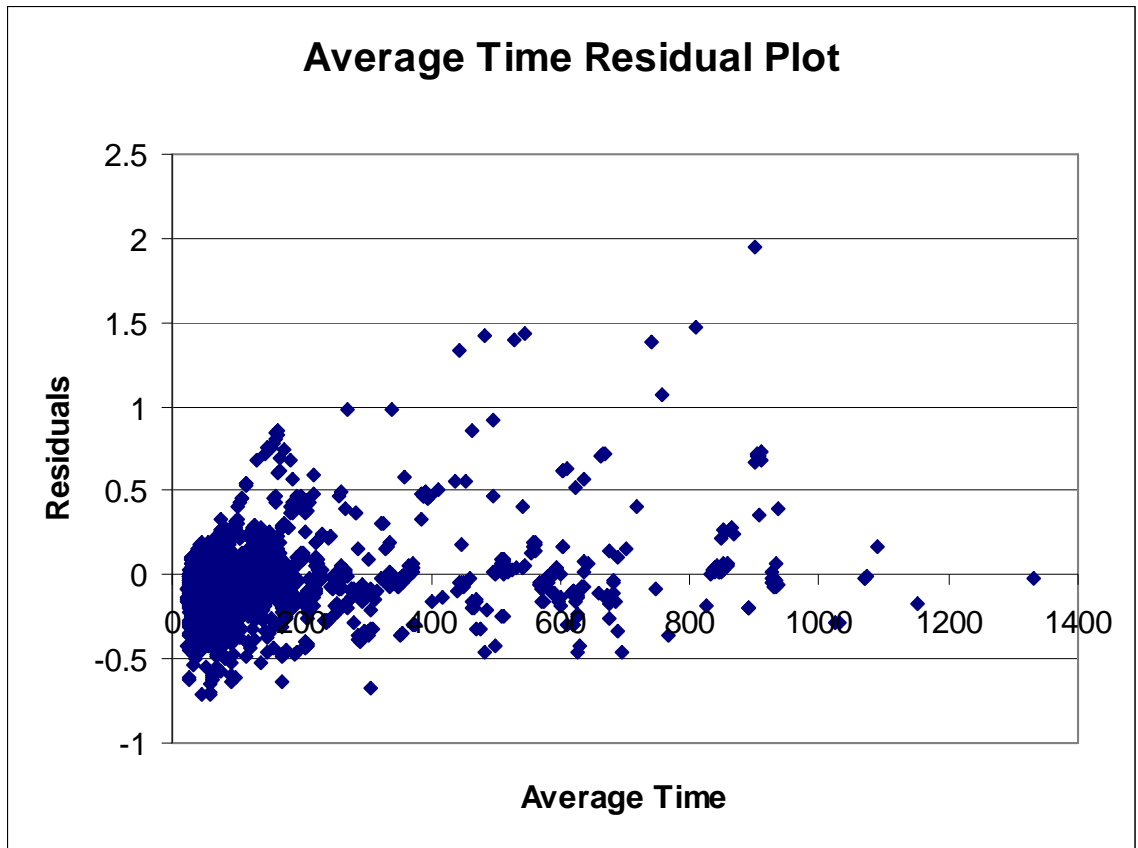


Figure 65: Boyen-Koller over SPI Average Speed Exponential Regression Model 2 Residuals Plot

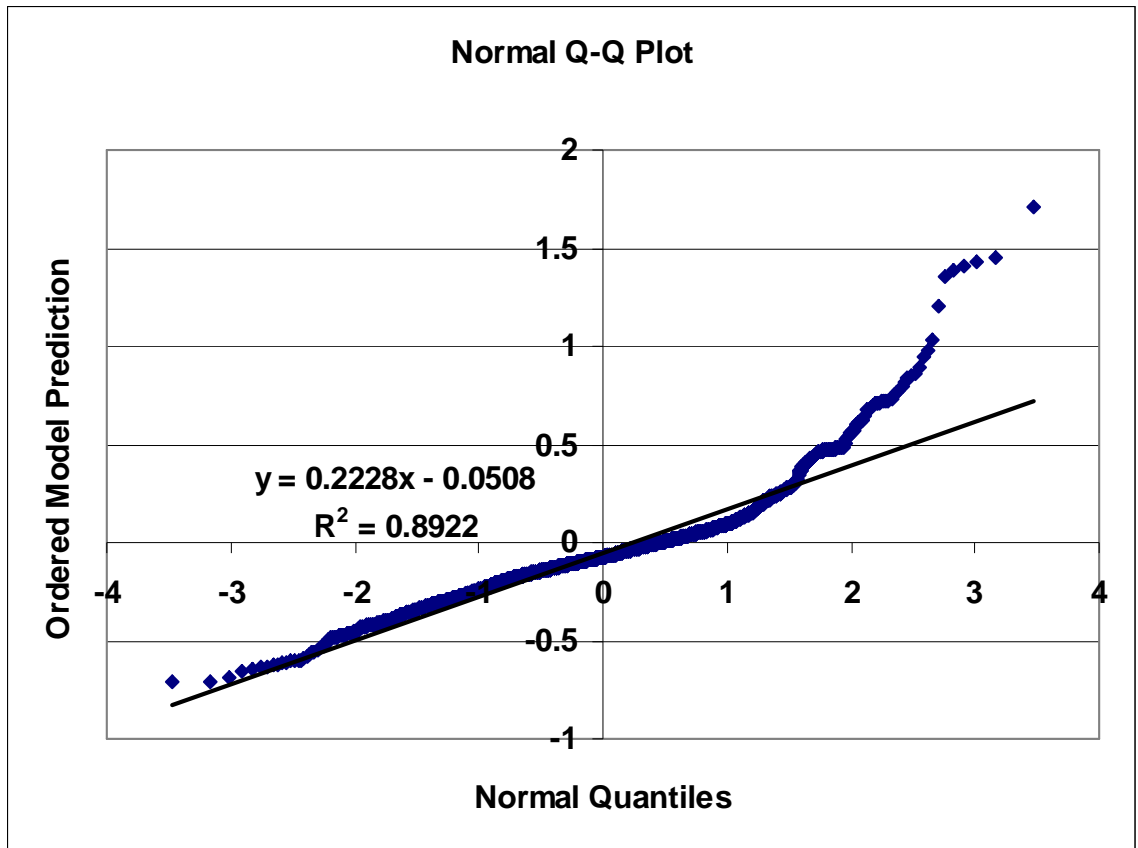


Figure 66: Boyen-Koller over SPI Average Speed Exponential Regression Model 2 Q-Q Plot

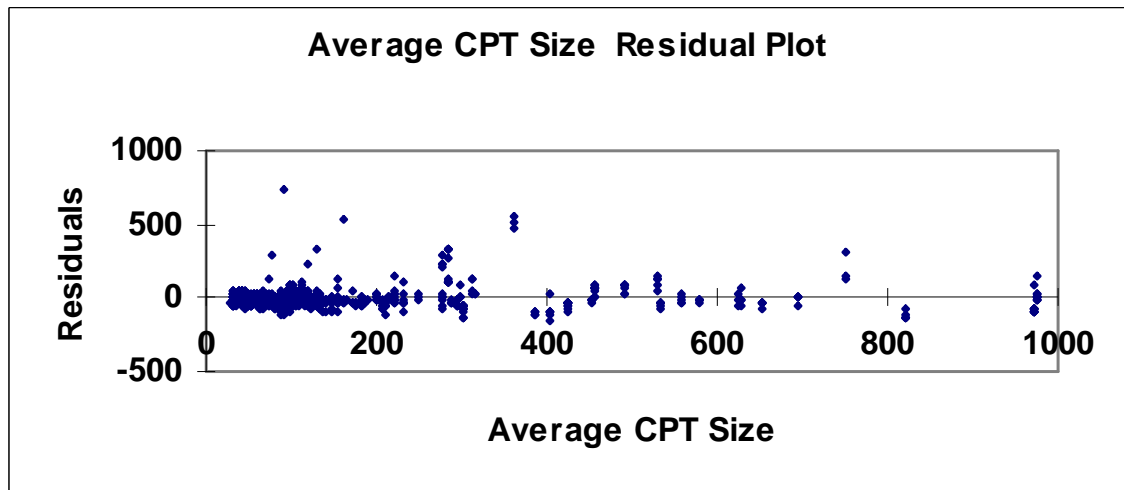


Figure 67: Boyen-Koller over SPI Average Speed Linear Regression Model 5 Residuals versus Average CPT Size Graph

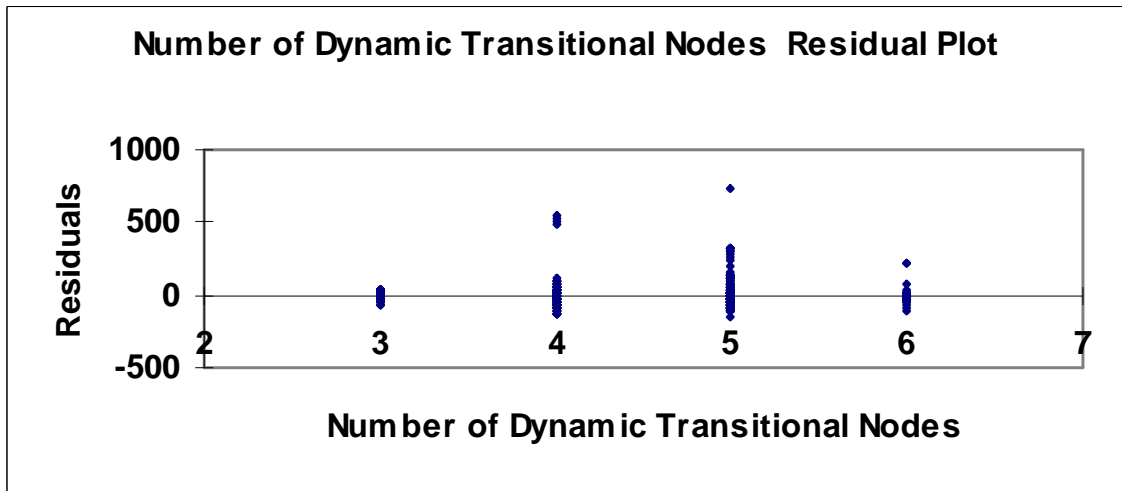


Figure 68: Boyen-Koller over SPI Average Speed Linear Regression Model 5 Residuals versus Number of Dynamic Transitional Nodes Graph

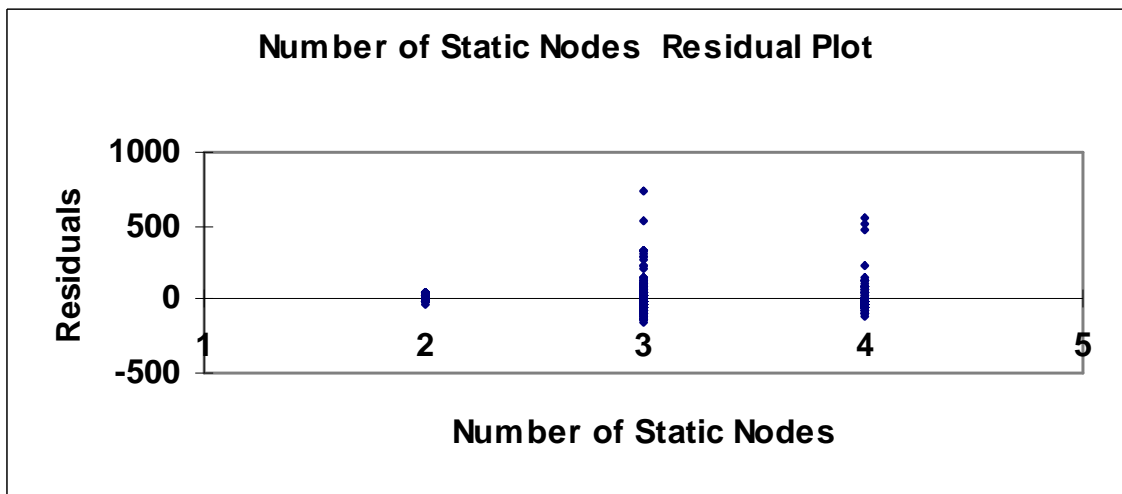


Figure 69: Boyen-Koller over SPI Average Speed Linear Regression Model 5 Residuals versus Number of Static Nodes Graph

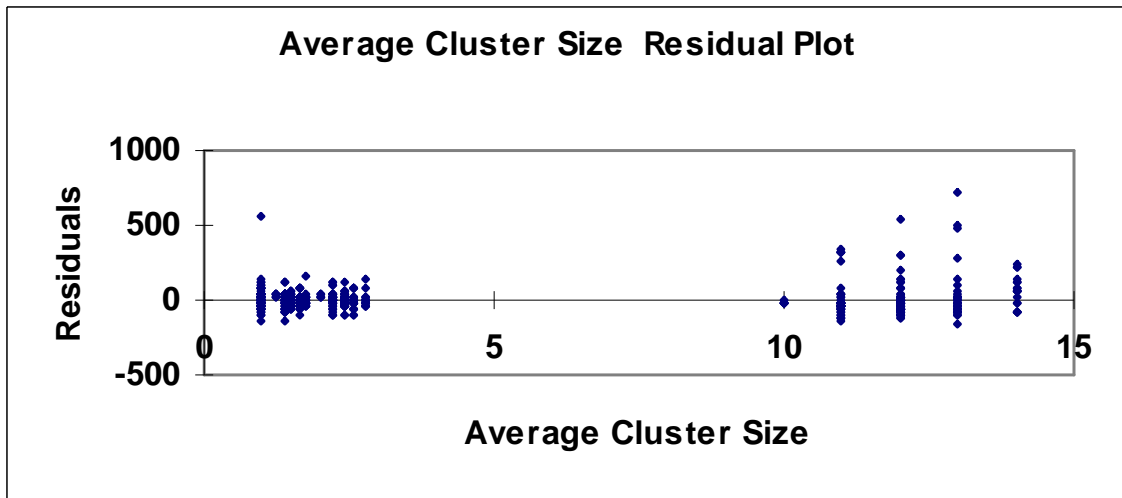


Figure 70: Boyen-Koller over SPI Average Speed Linear Regression Model 5 Residuals versus Average Cluster Size Graph

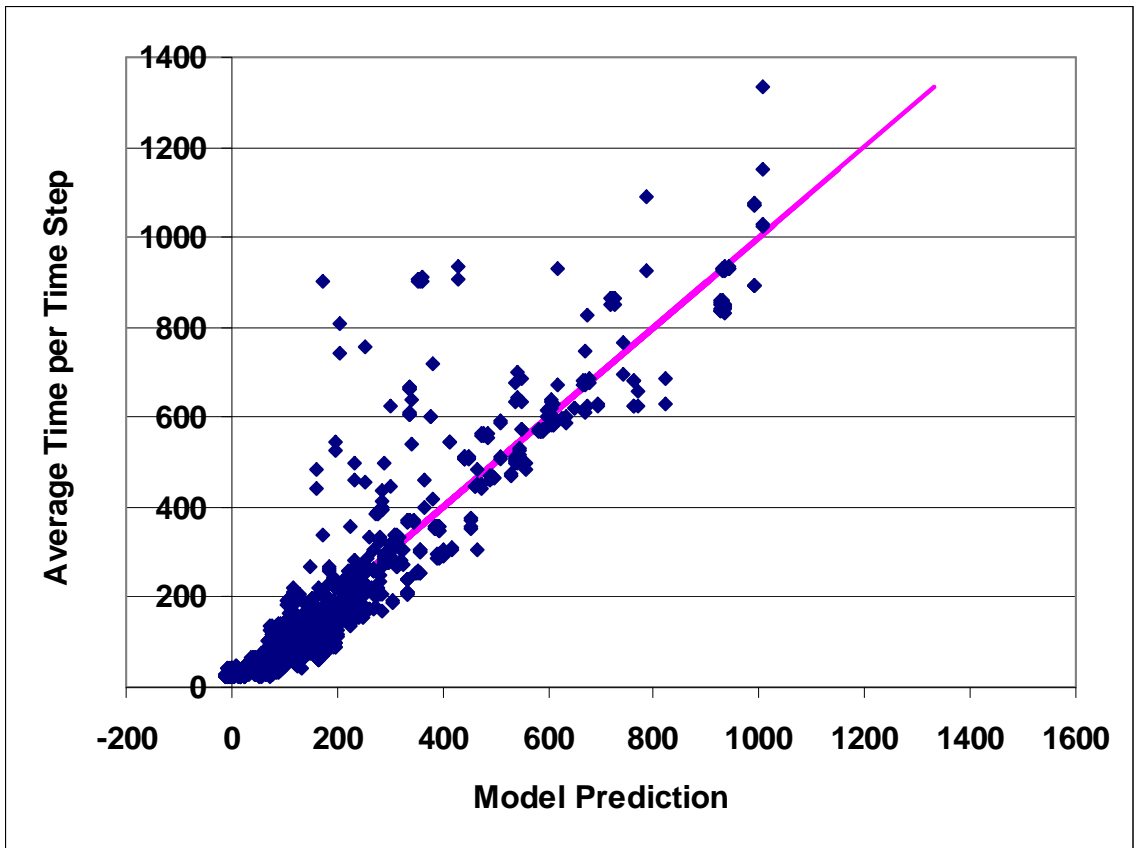


Figure 71: Boyen-Koller over SPI Average Speed Linear Regression Model 5 Predicted versus Actual Graph

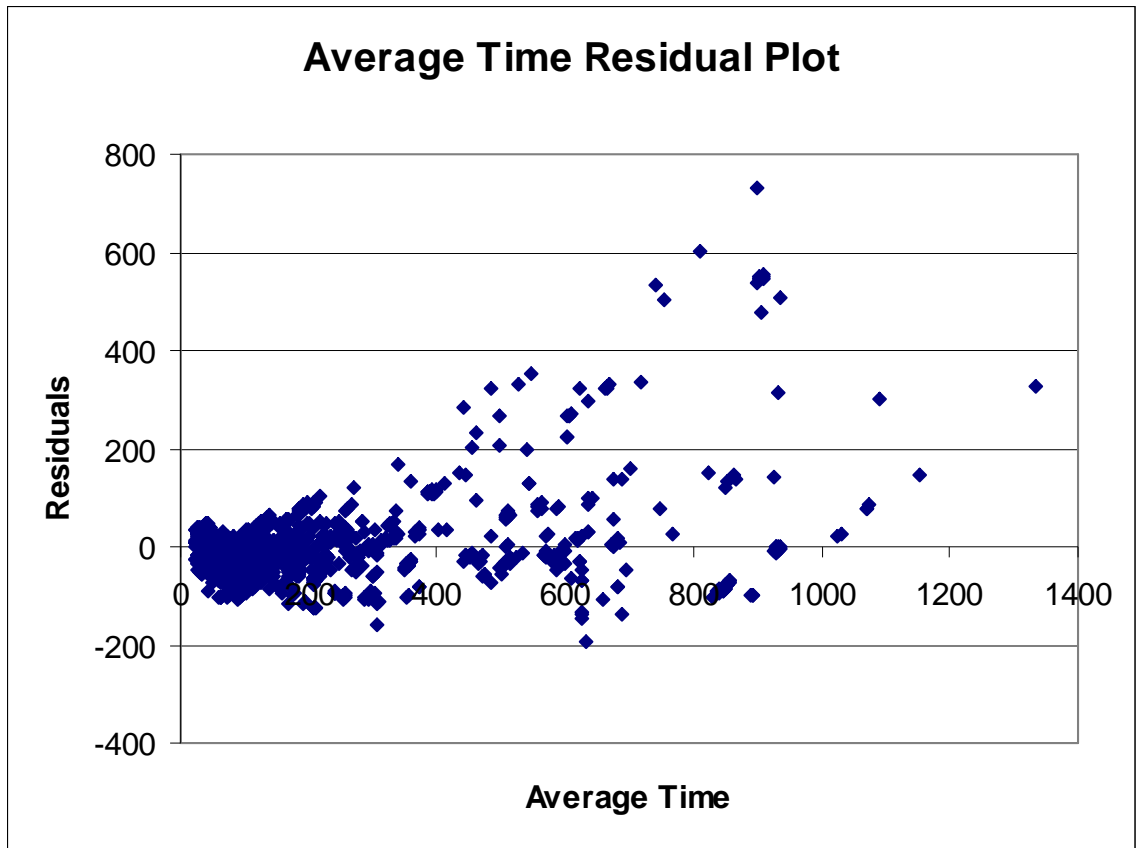


Figure 72: Boyen-Koller over SPI Average Speed Linear Regression Model 5 Residuals Plot

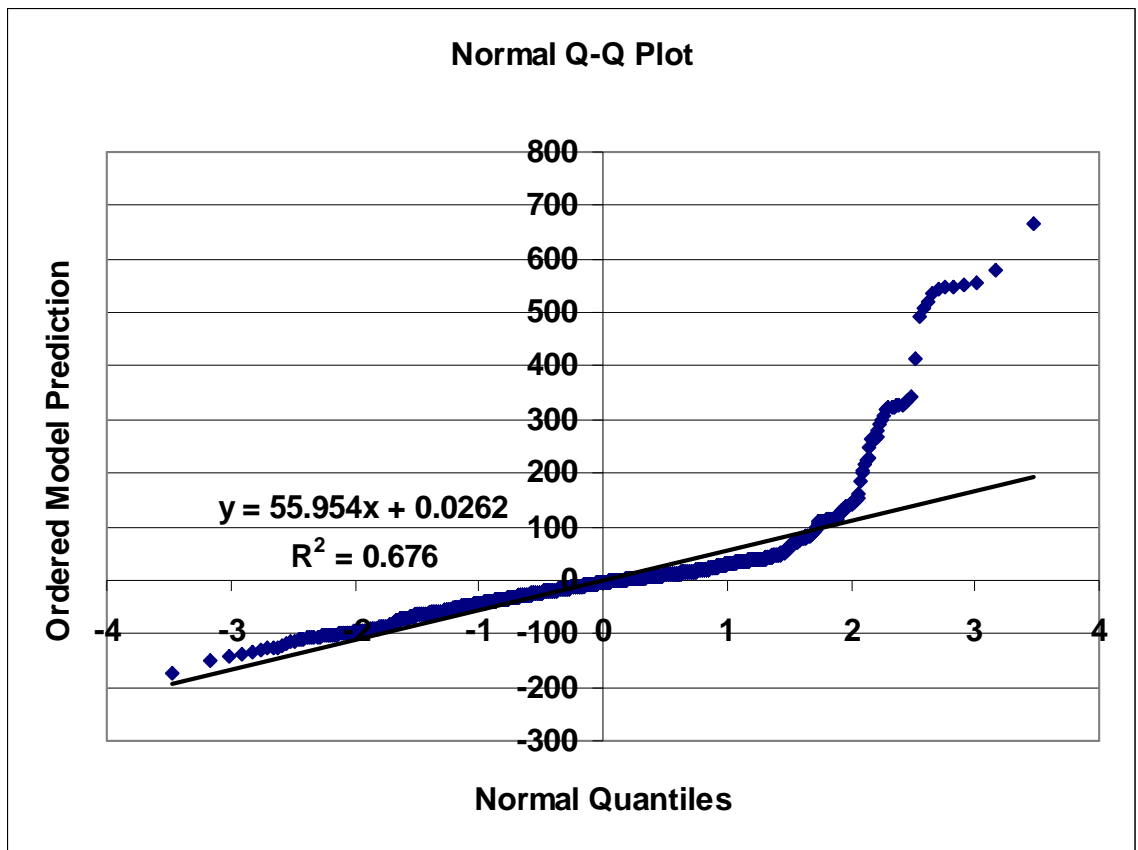


Figure 73: Boyen-Koller over SPI Average Speed Linear Regression Model 5 Q-Q Plot

Appendix D: Regression Diagnostics for the Boyen-Koller Variance of Speed of
Inference Models

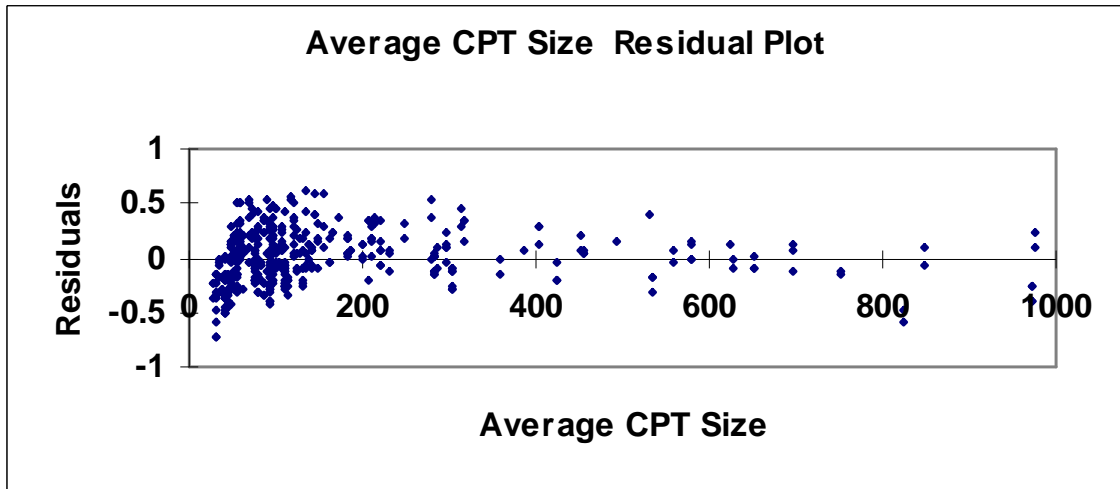


Figure 74: Boyen-Koller over SPI Variance of Speed Exponential Regression Model 1A Residuals versus Average CPT Size Graph

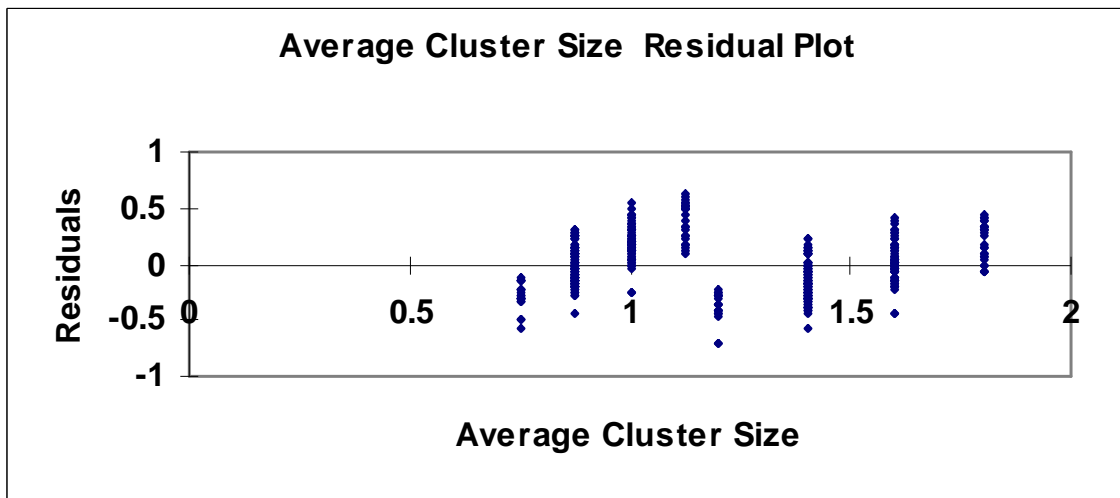


Figure 75: Boyen-Koller over SPI Variance of Speed Exponential Regression Model 1A Residuals versus Average Cluster Size Graph

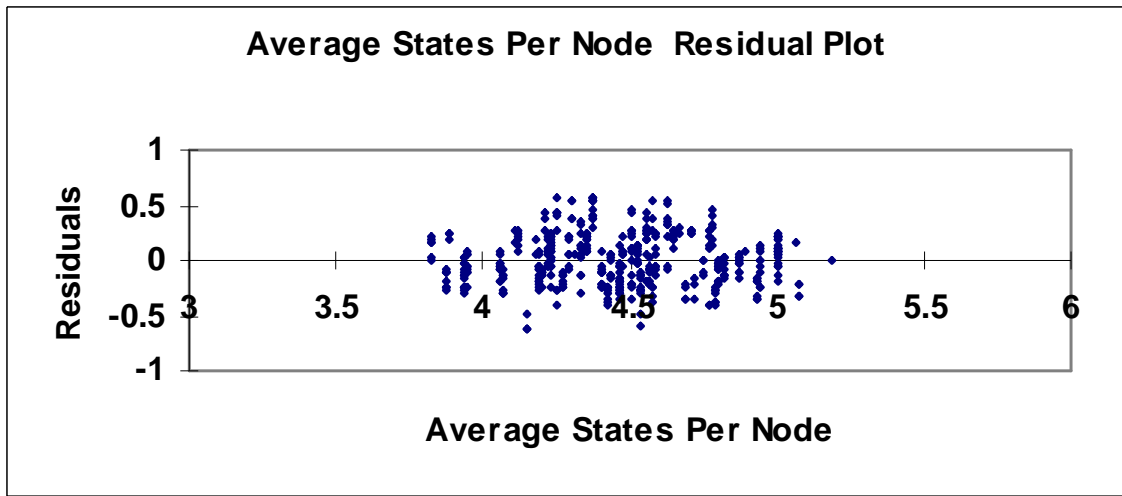


Figure 76: Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2A Residuals versus Average States per Node Graph

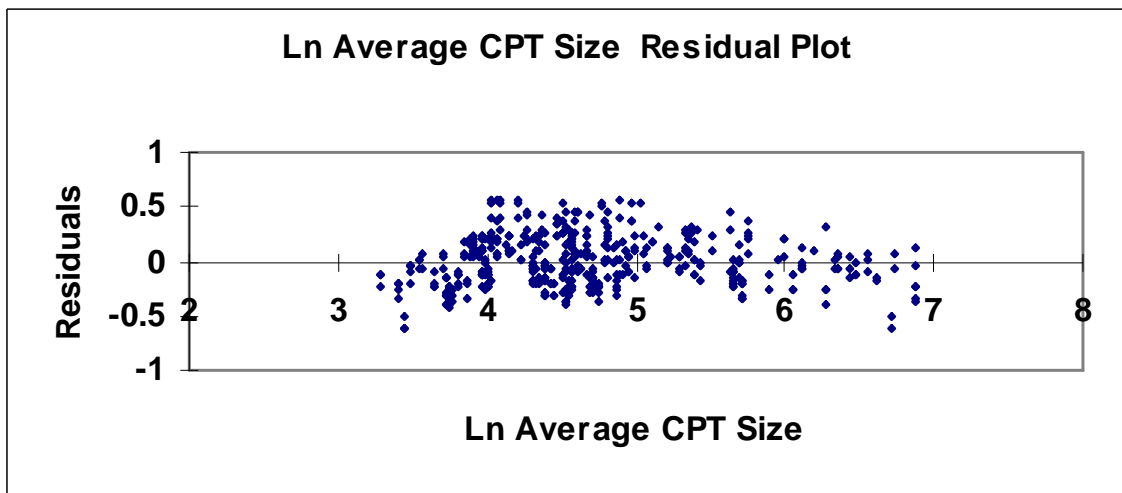


Figure 77: Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2A Residuals versus the Natural Logarithm of the Average CPT Size Graph

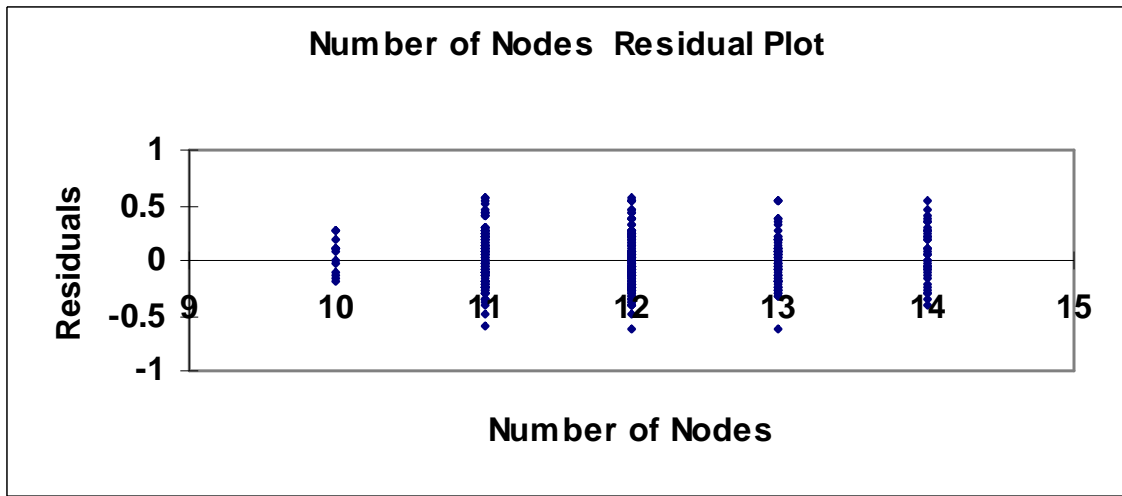


Figure 78: Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2A Residuals versus Number of Nodes Graph

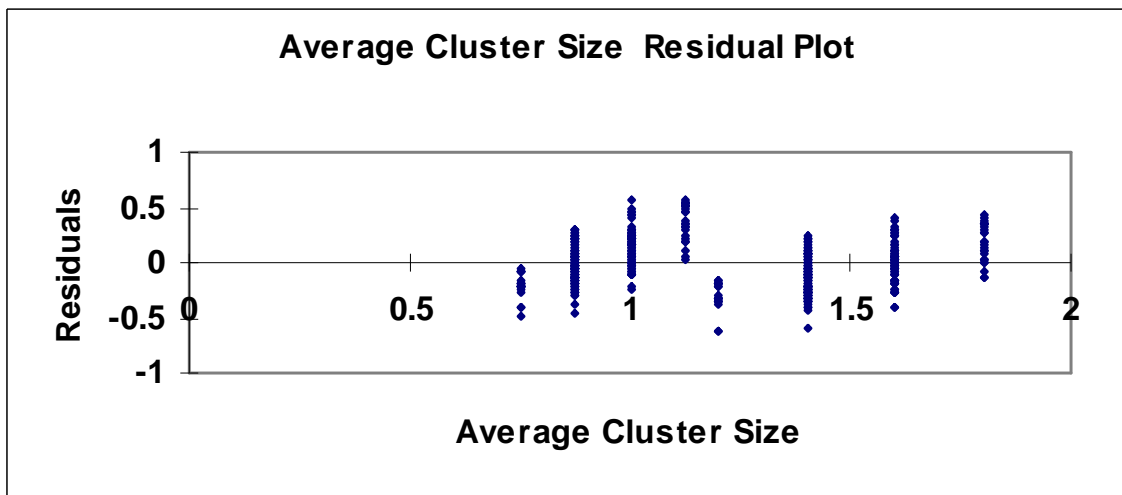


Figure 79: Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2A Residuals versus Average Cluster Size Graph

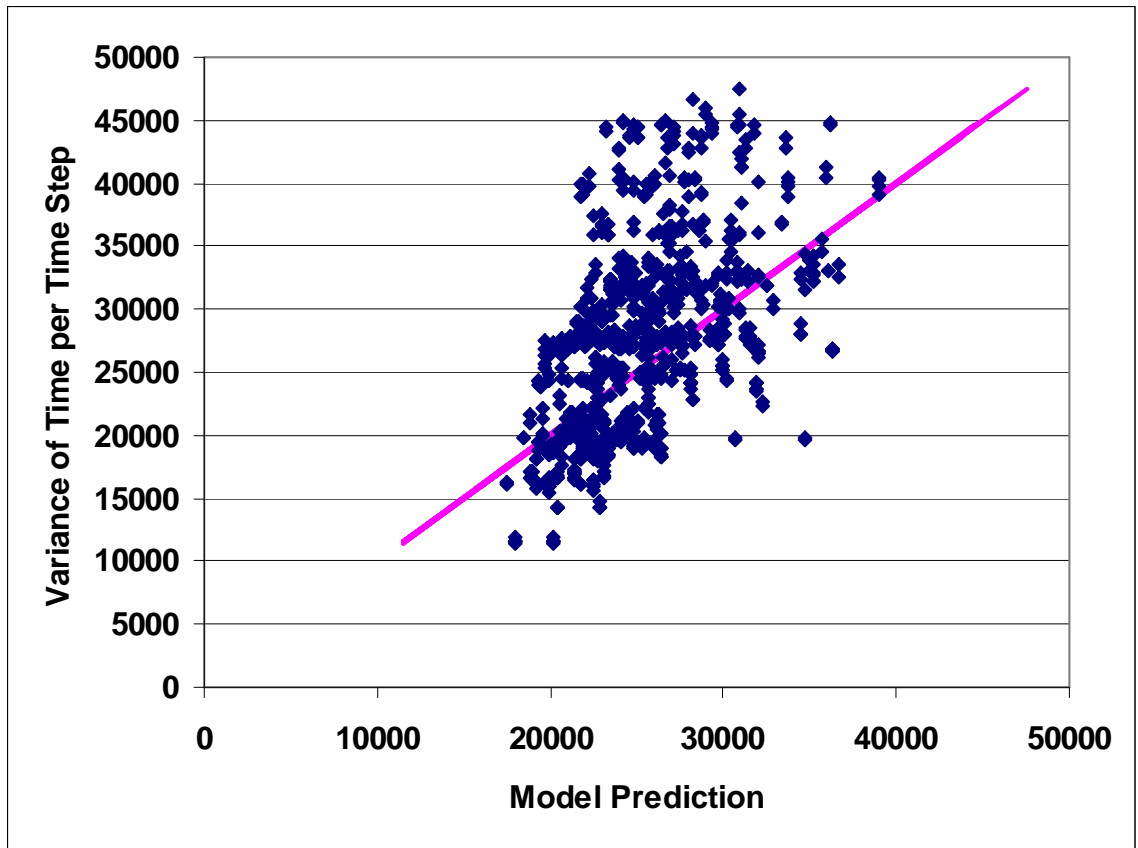


Figure 80: Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2A Predicted versus Actual Graph

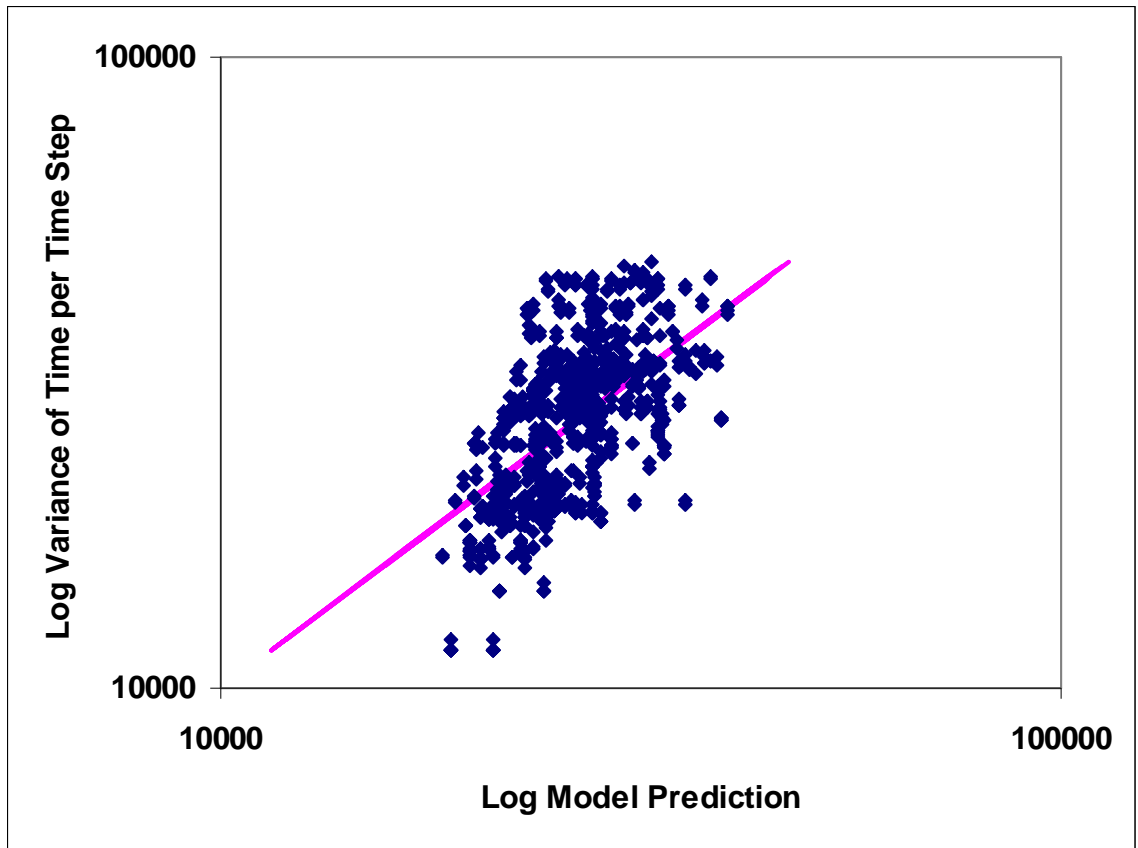


Figure 81: Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2A Predicted versus Actual Graph in Logarithm Space

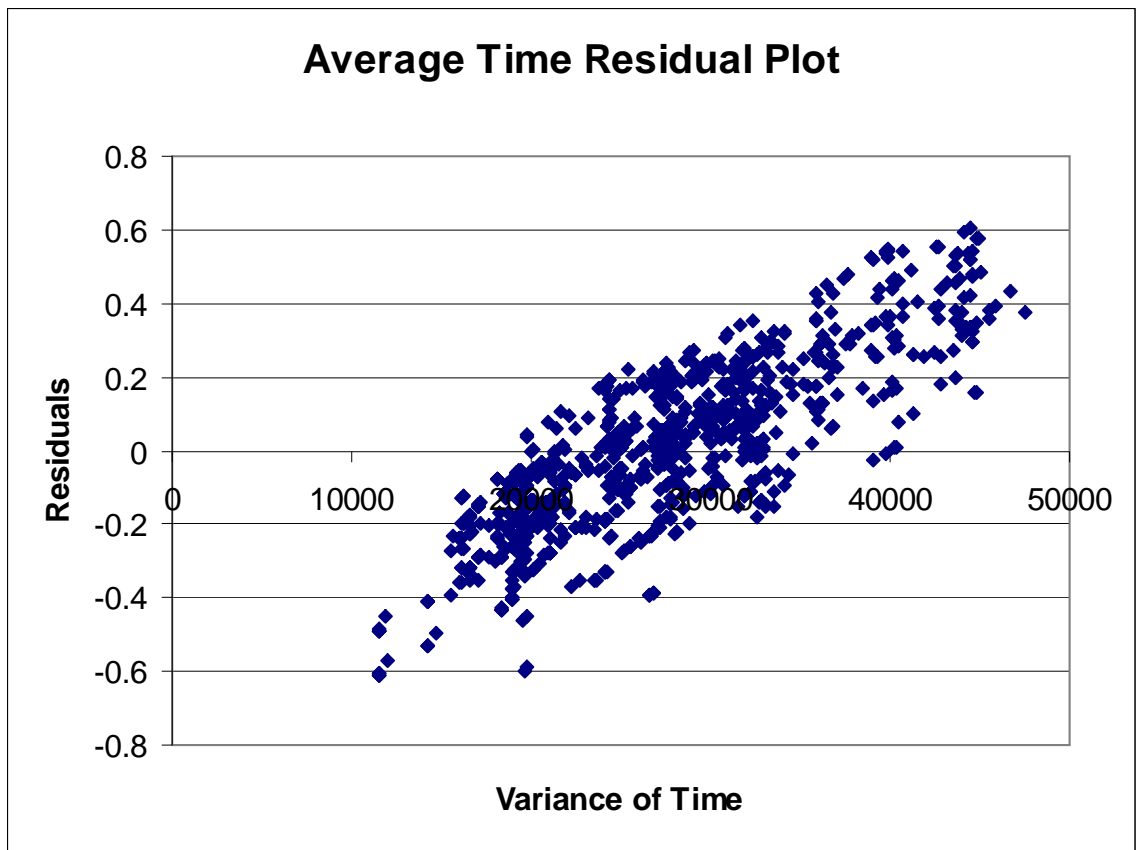


Figure 82: Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2A Residuals Plot

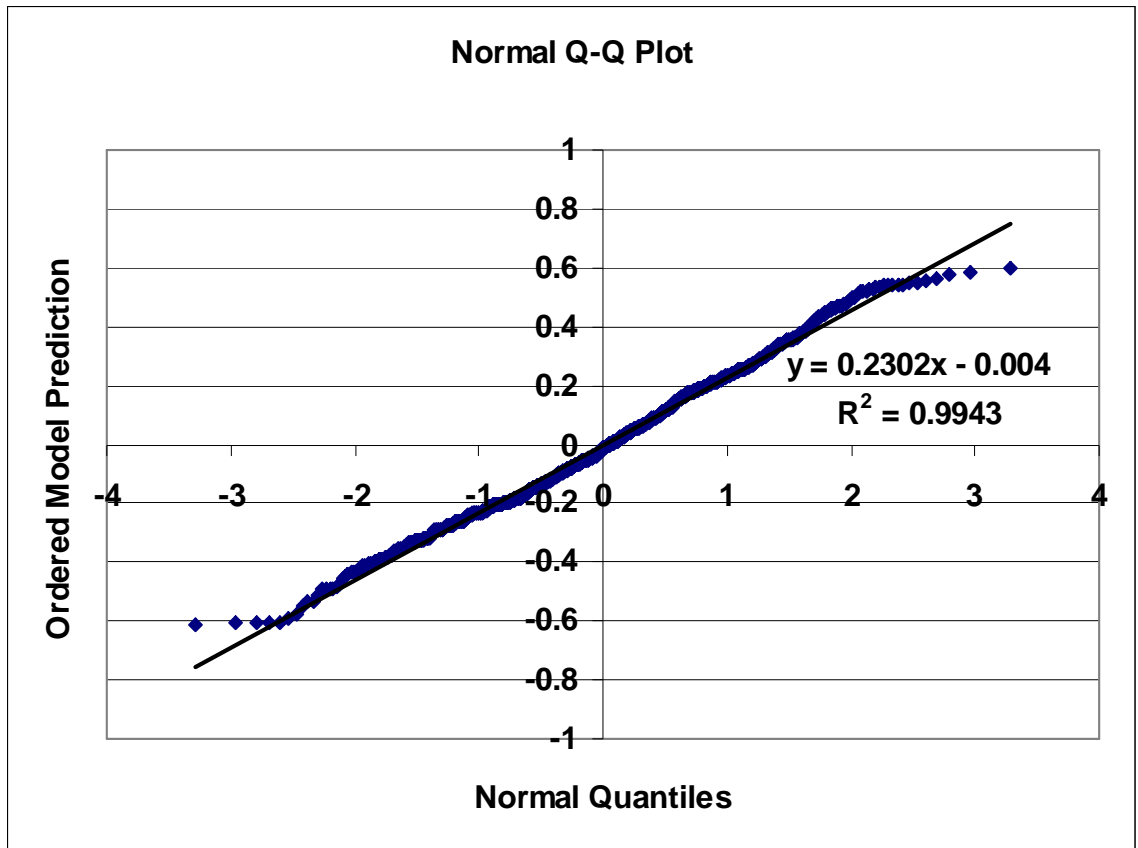


Figure 83: Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2A Q-Q Plot

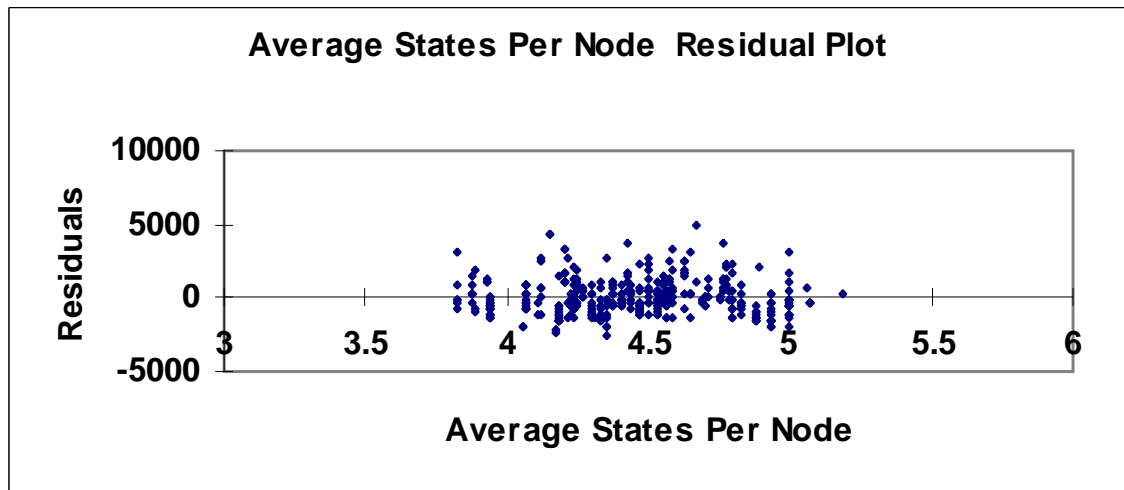


Figure 84: Boyen-Koller over SPI Variance of Speed Linear Regression Model 4A Residuals versus Average States per Node Graph

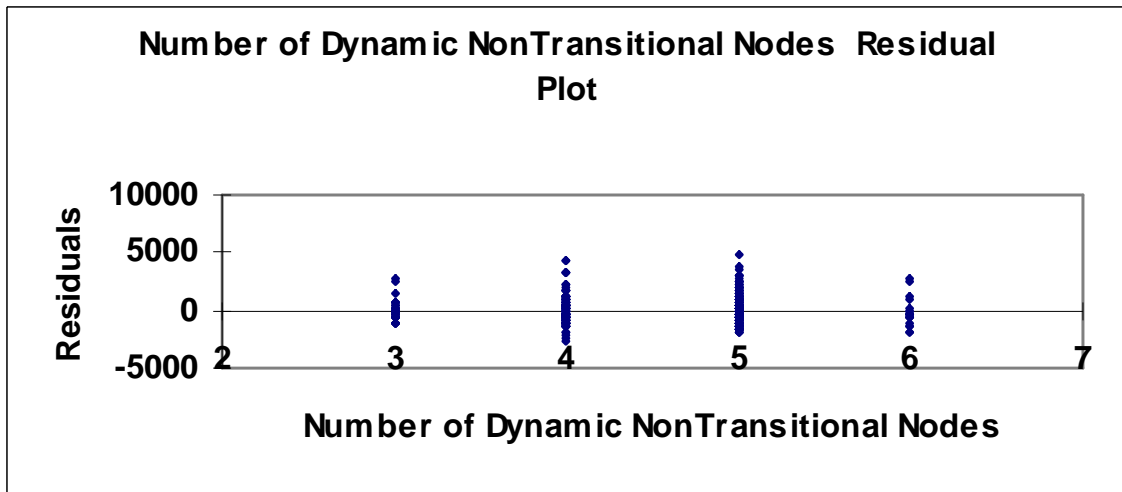


Figure 85: Boyen-Koller over SPI Variance of Speed Linear Regression Model 4A Residuals versus Number of Dynamic NonTransitional Nodes Graph

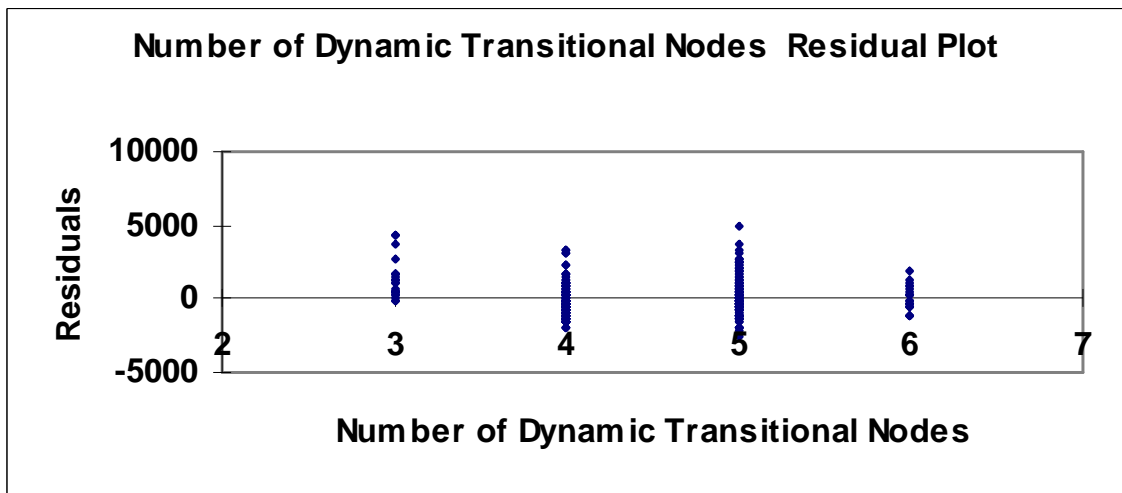


Figure 86: Boyen-Koller over SPI Variance of Speed Linear Regression Model 4A Residuals versus Number of Dynamic Transitional Nodes Graph

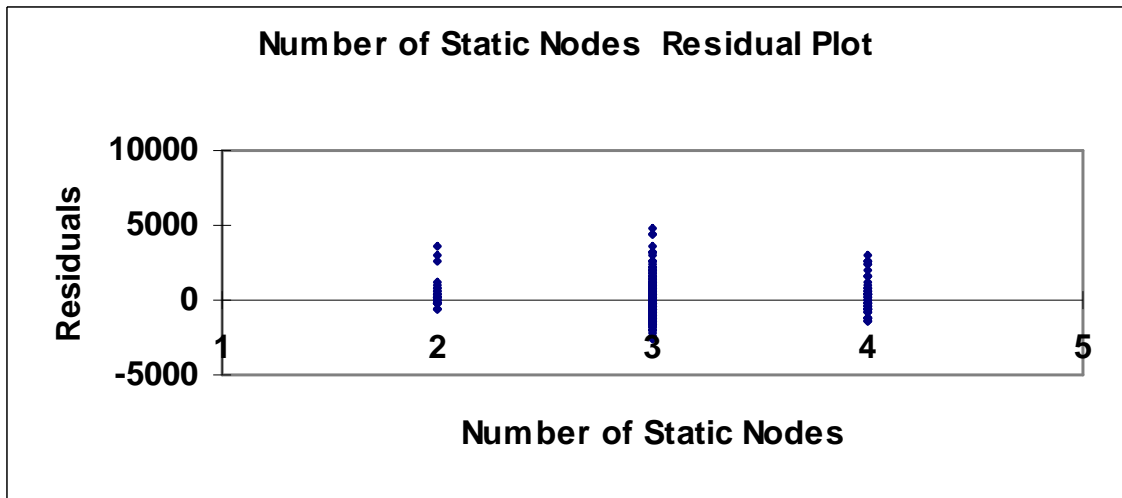


Figure 87: Boyen-Koller over SPI Variance of Speed Linear Regression Model 4A Residuals versus Number of Static Nodes Graph

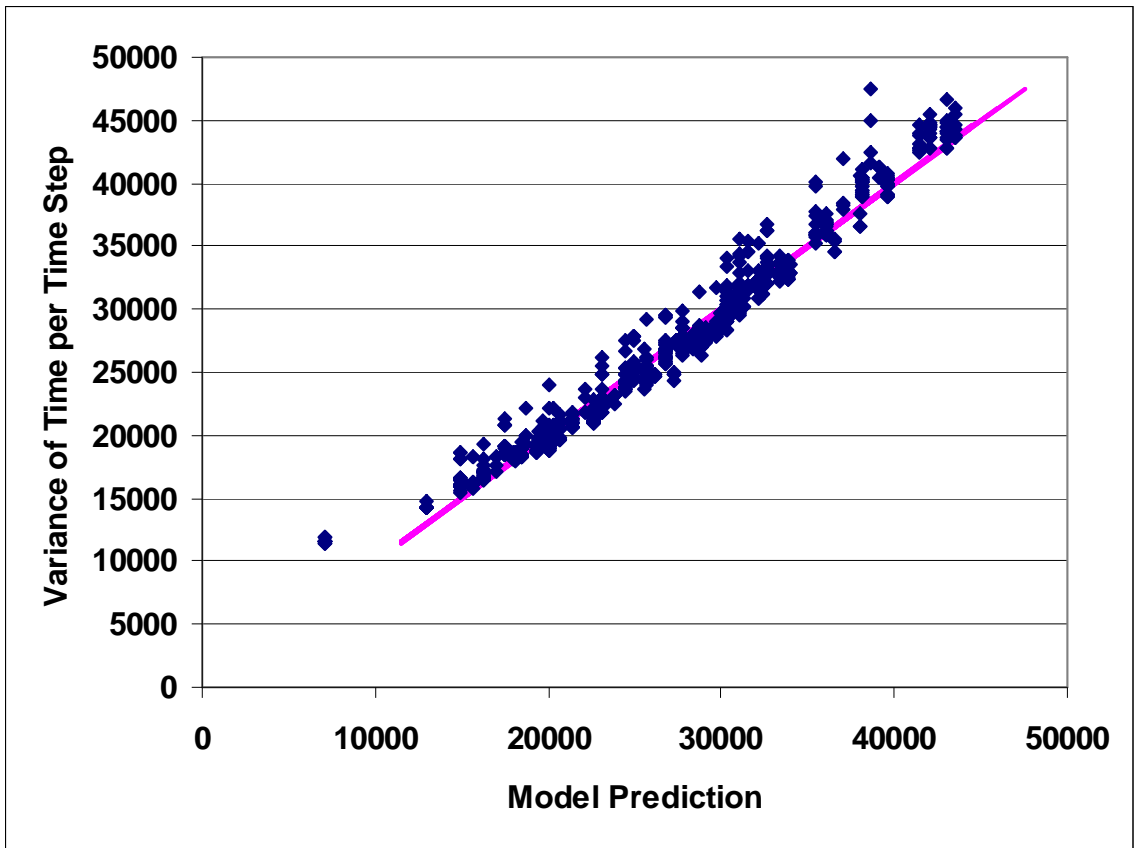


Figure 88: Boyen-Koller over SPI Variance of Speed Linear Regression Model 4A Predicted versus Actual Graph

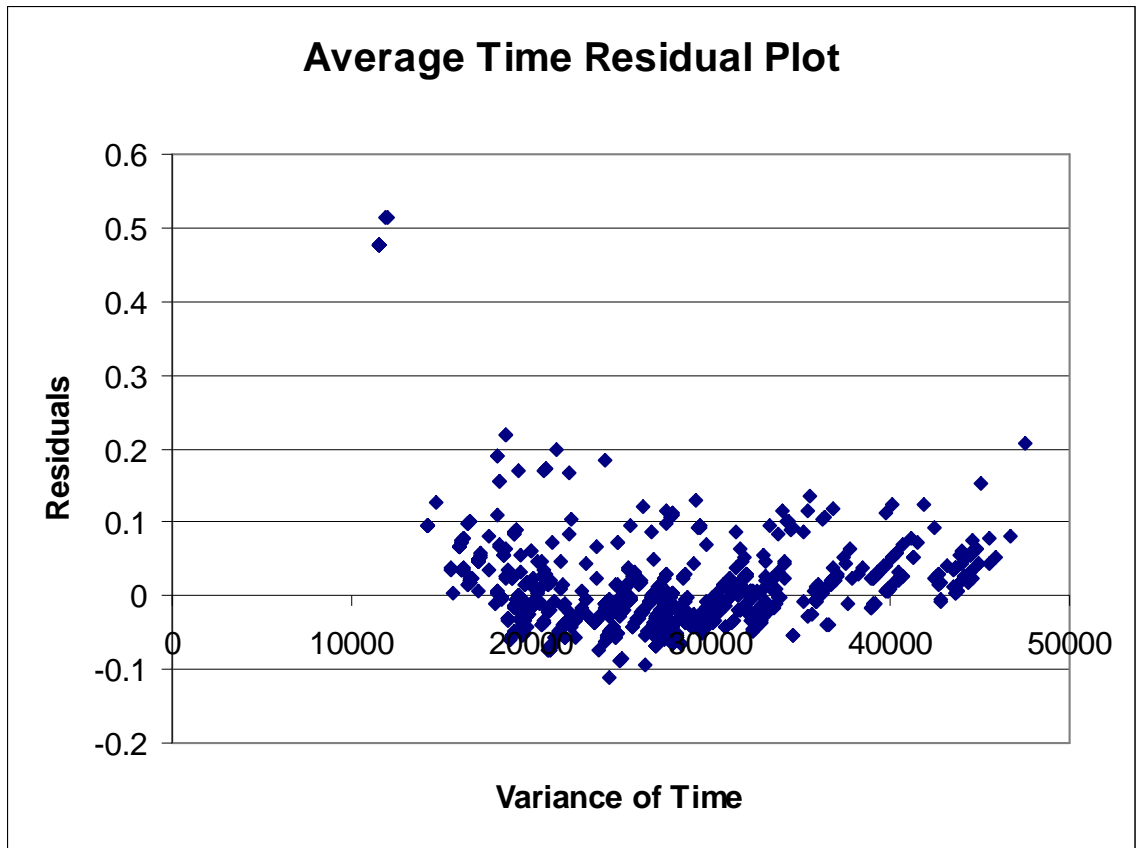


Figure 89: Boyen-Koller over SPI Variance of Speed Linear Regression Model 4A Residuals Plot

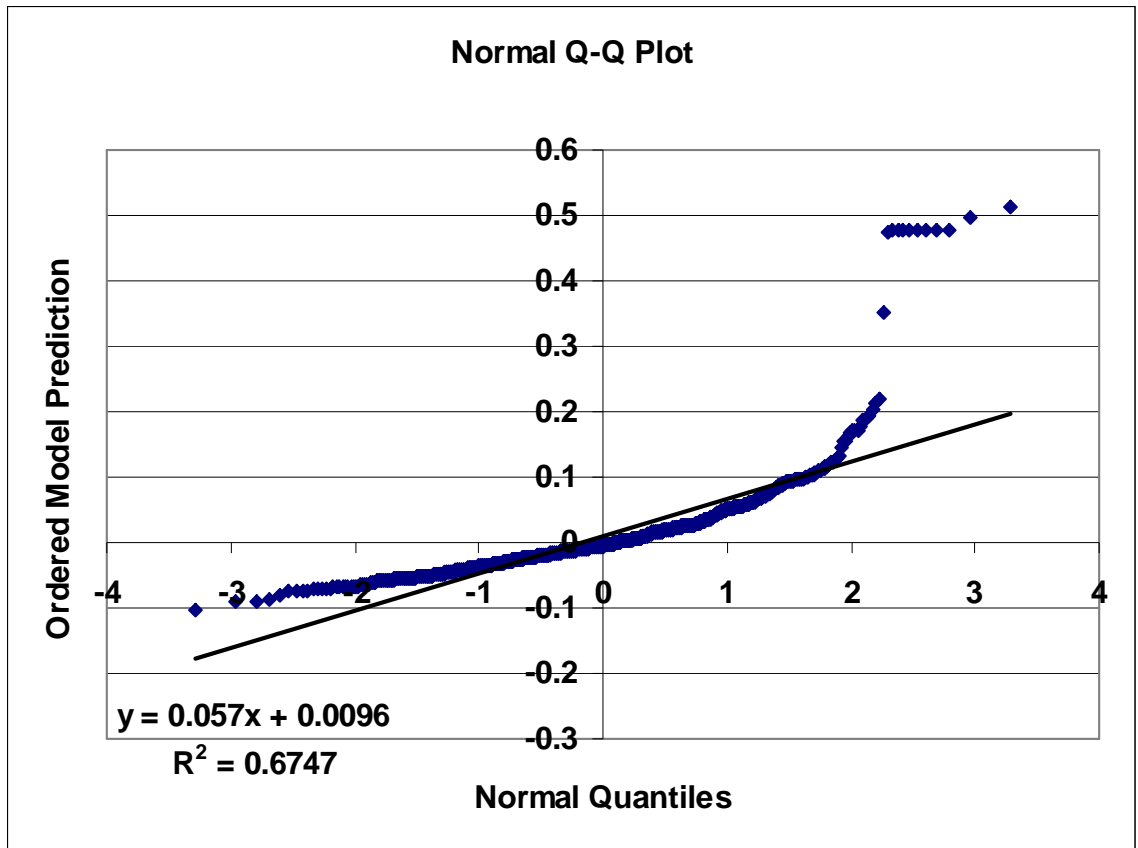


Figure 90: Boyen-Koller over SPI Variance of Speed Linear Regression Model 4A Q-Q Plot

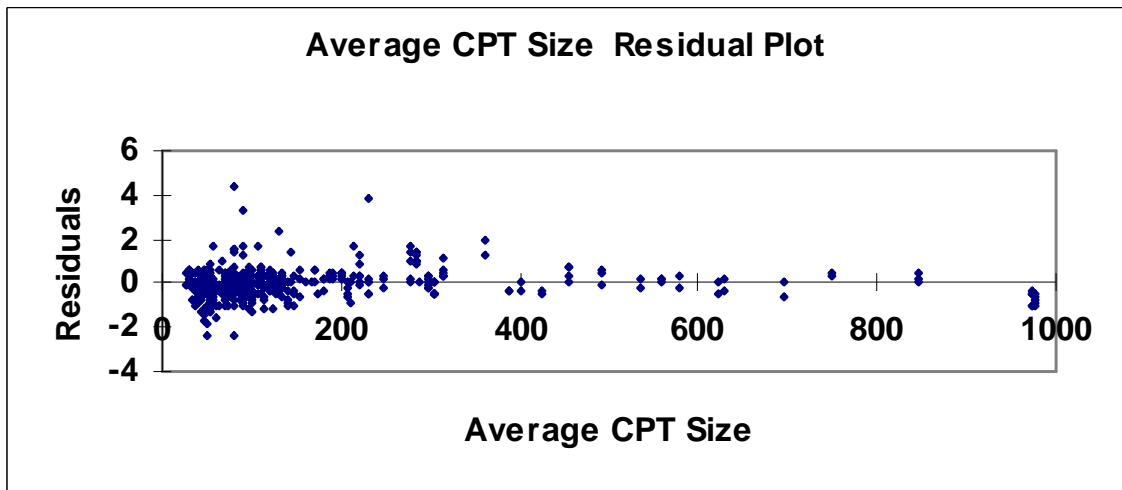


Figure 91: Boyen-Koller over SPI Variance of Speed Exponential Regression Model 1B Residuals versus Average CPT Size Graph

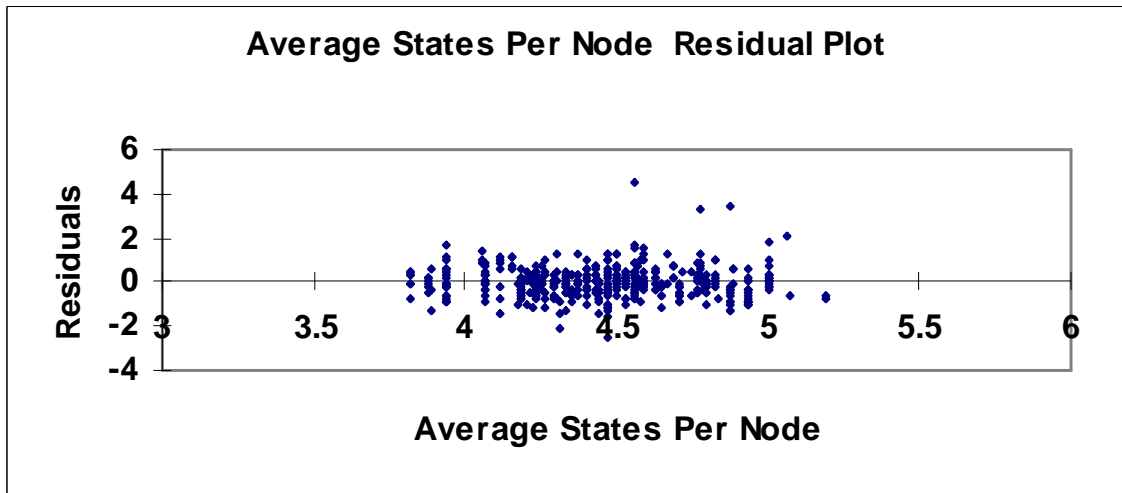


Figure 92: Boyen-Koller over SPI Variance of Speed Linear Regression Model 2B Residuals versus Average States per Node Graph

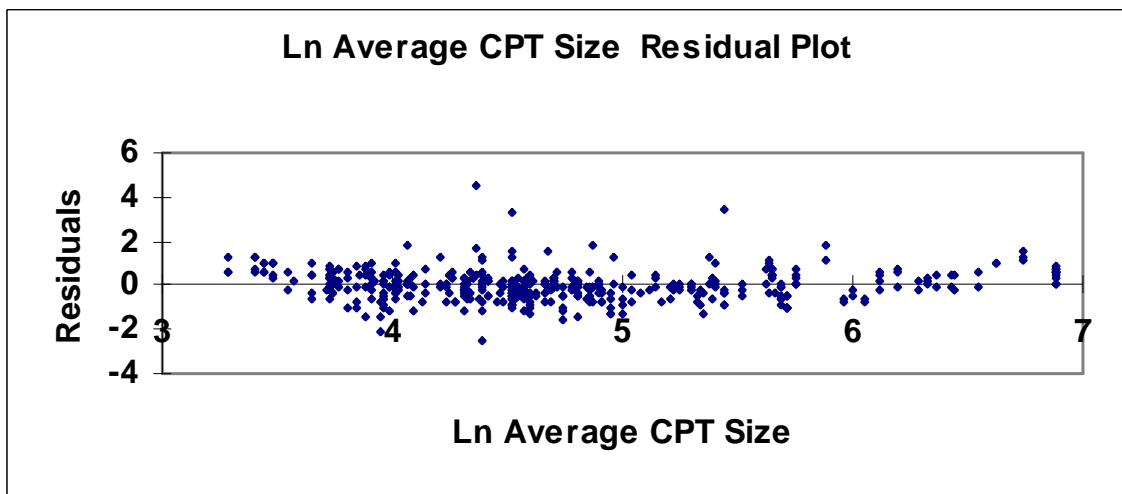


Figure 93: Boyen-Koller over SPI Variance of Speed Linear Regression Model 2B Residuals versus the Natural Logarithm of the Average CPT Size Graph

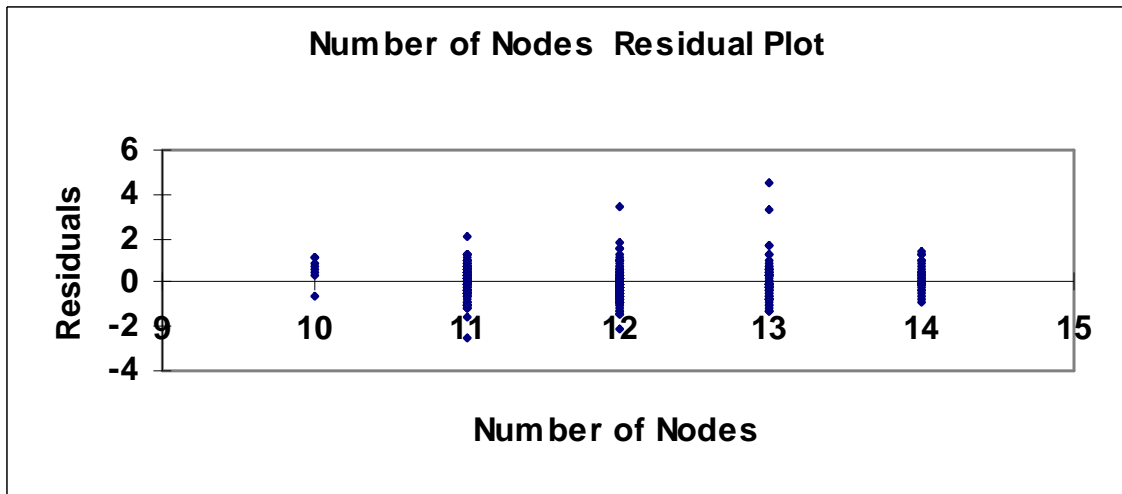


Figure 94: Boyen-Koller over SPI Variance of Speed Linear Regression Model 2B Residuals versus Number of Nodes Graph

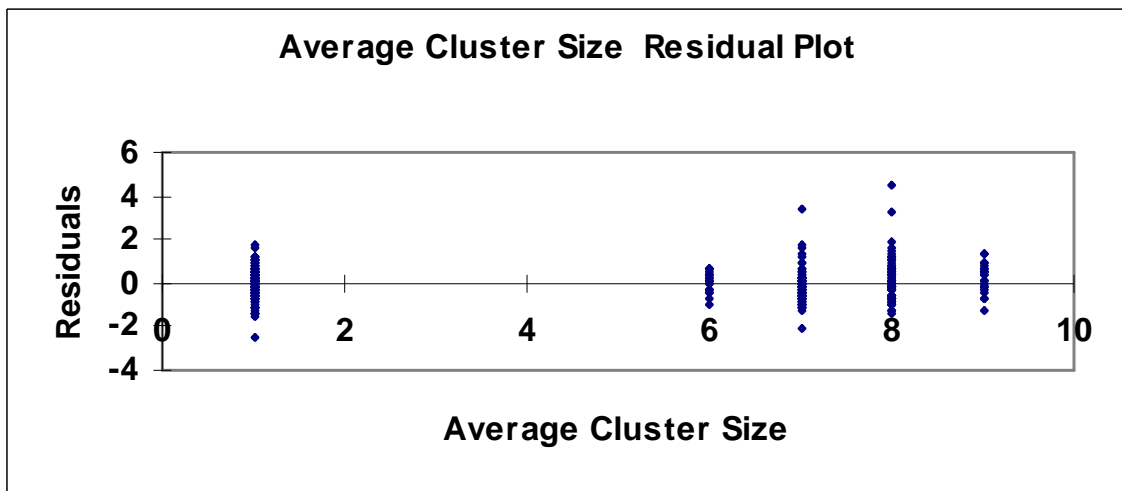


Figure 95: Boyen-Koller over SPI Variance of Speed Linear Regression Model 2B Residuals versus Average Cluster Size Graph

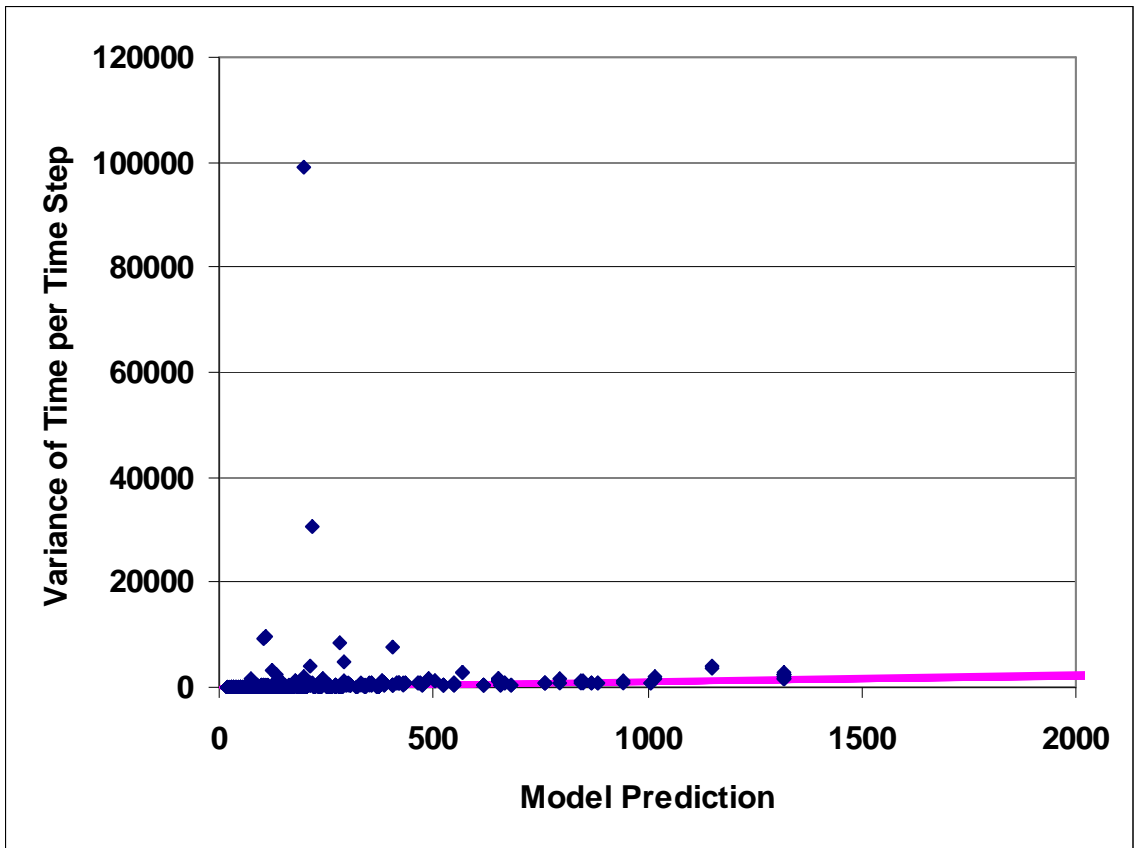


Figure 96: Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2B Predicted versus Actual Graph

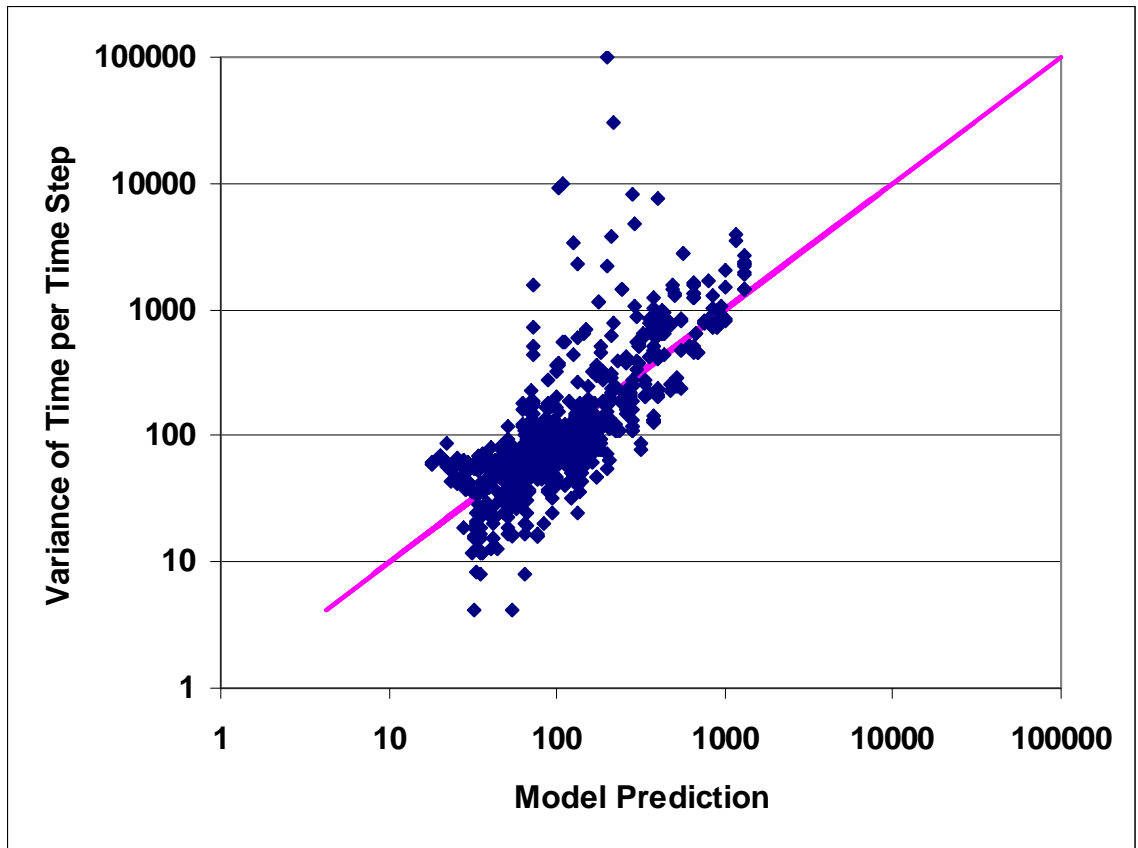


Figure 97: Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2B Predicted versus Actual Graph in Logarithm Space

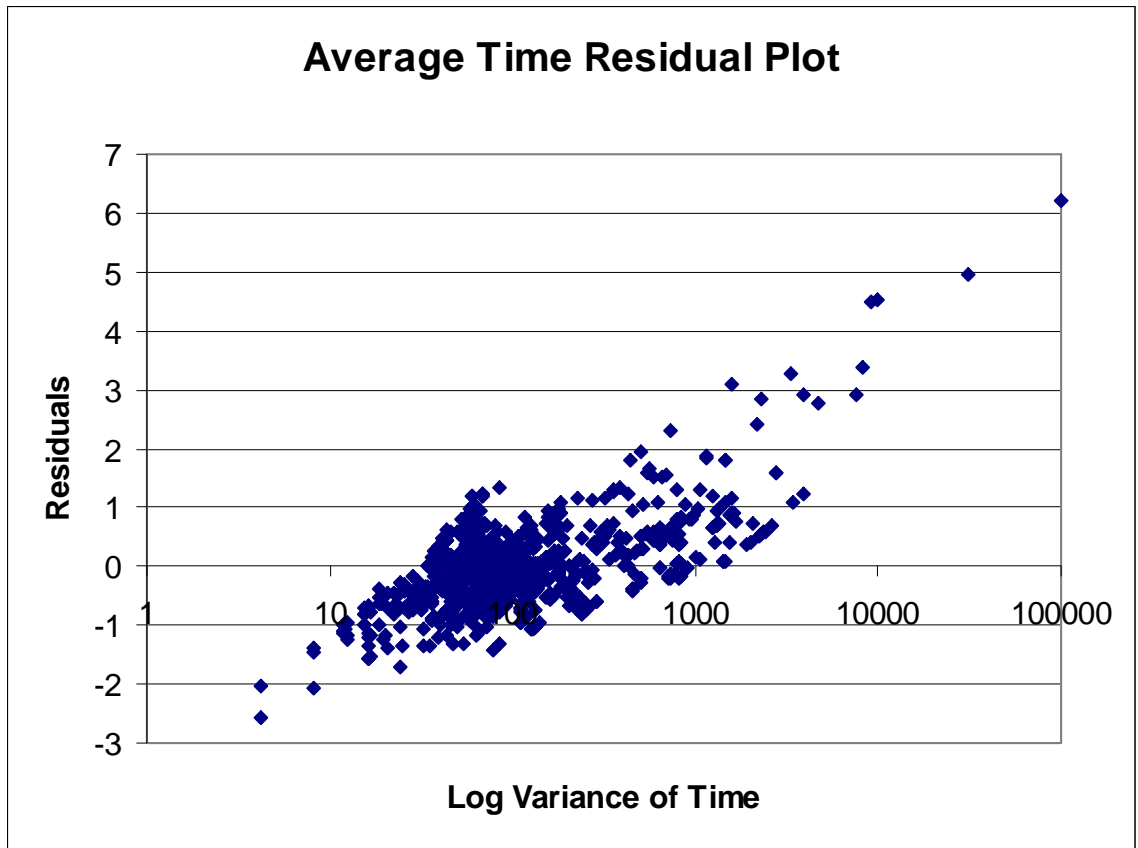


Figure 98: Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2B Residuals Plot

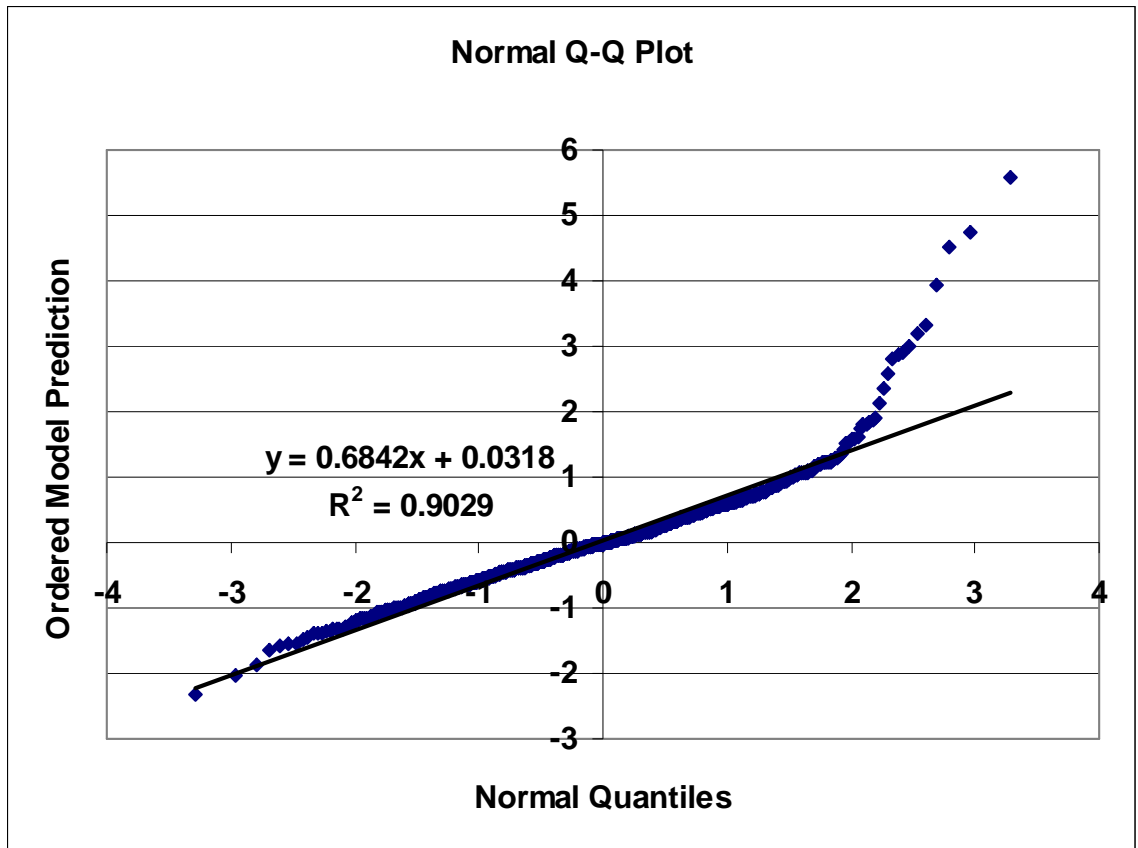


Figure 99: Boyen-Koller over SPI Variance of Speed Exponential Regression Model 2B Q-Q Plot

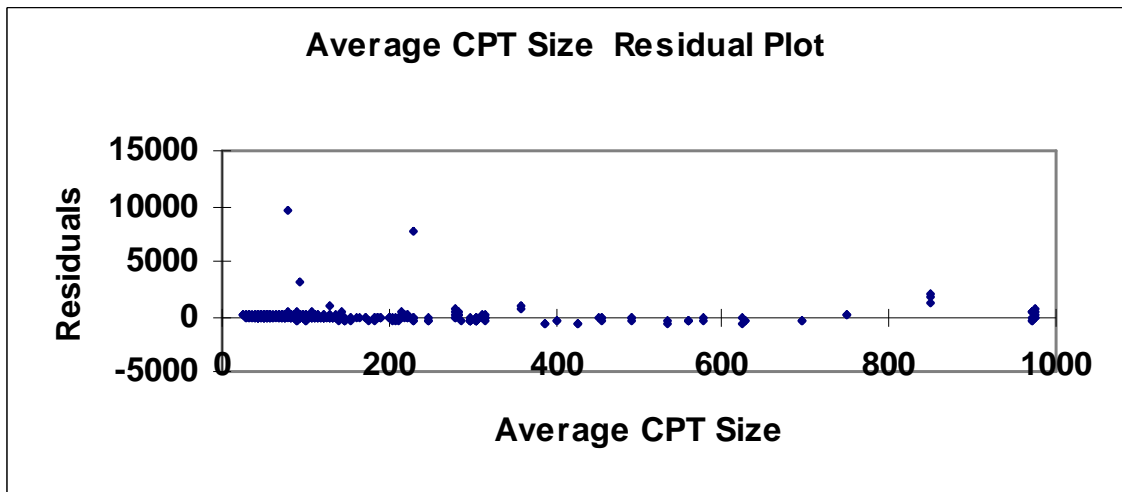


Figure 100: Boyen-Koller over SPI Variance of Speed Exponential Regression Model 1B Residuals versus Average CPT Size Graph

Appendix E: Random Partially Dynamic Bayesian Network Generator

The primary function of the Random PDBN Generator is to generate random PDBNs for the purpose of studying various inference algorithms for dynamic Bayesian networks. The underlying impetus behind the Random PDBN Generator is to generate a large number of disparate dynamic Bayesian networks quickly in such a way that the distribution of the features of these Bayesian networks is predictable and controllable. At the same time, this software is intended to be able to be used in a generalized set of situations other than the purposes of this research study.

I have summarized the functionality the Random PDBN Generator provides in Figure 101. This figure is a use case diagram listing the functionality provided to the user of the random PDBN generator. In this diagram, the stick figure represents the user and the ovals represent the functionality available to the user of the random PDBN generator.

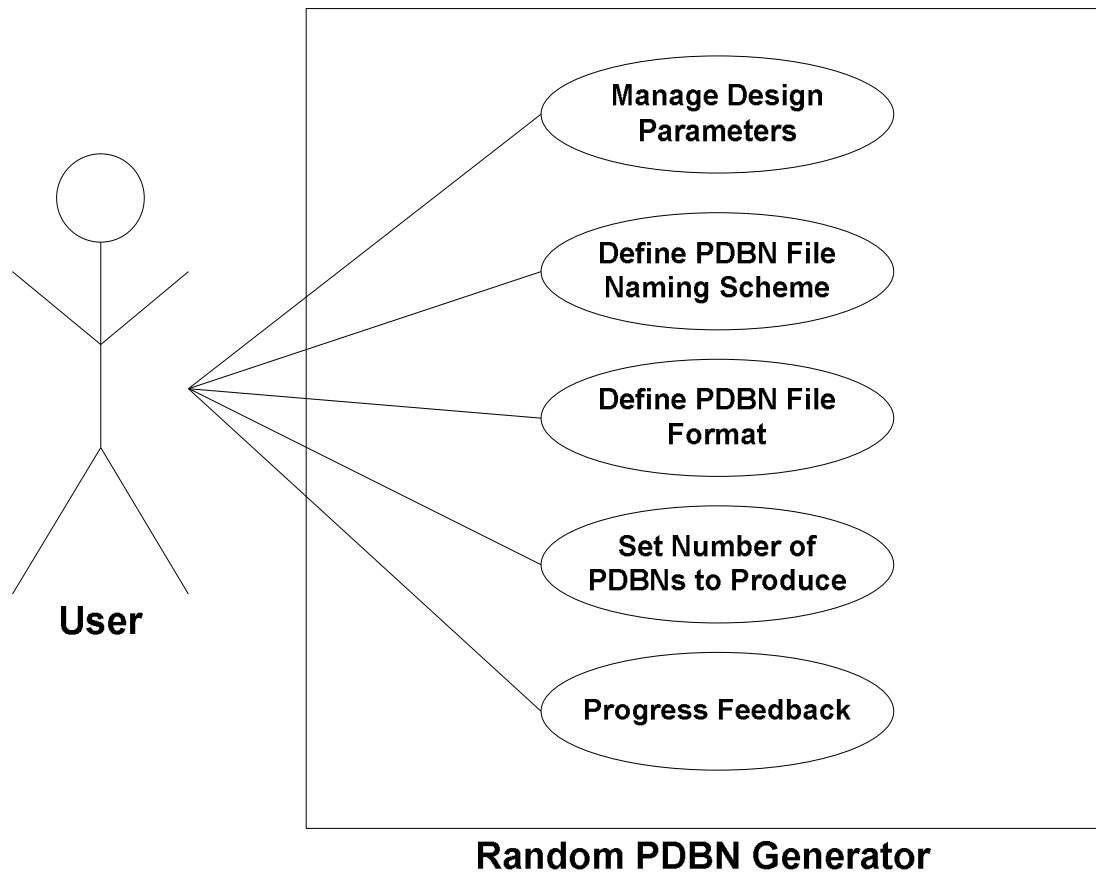


Figure 101: Use Case Diagram for the Random PDBN Generator

Figure 102 is a data flow diagram and it describes how the Random PDBN Generator might utilize data from the user to produce PDBNs for the software program testing the PDBNs generated. Here, the ovals represent processes performed by the software application, the rectangles represent actors, and the double-lined boxes represent data stores. The arcs between two elements represent the transfer of data with the direction of the arc representing the direction of that data's flow. The labels on each arc describes the data that is being transferred.

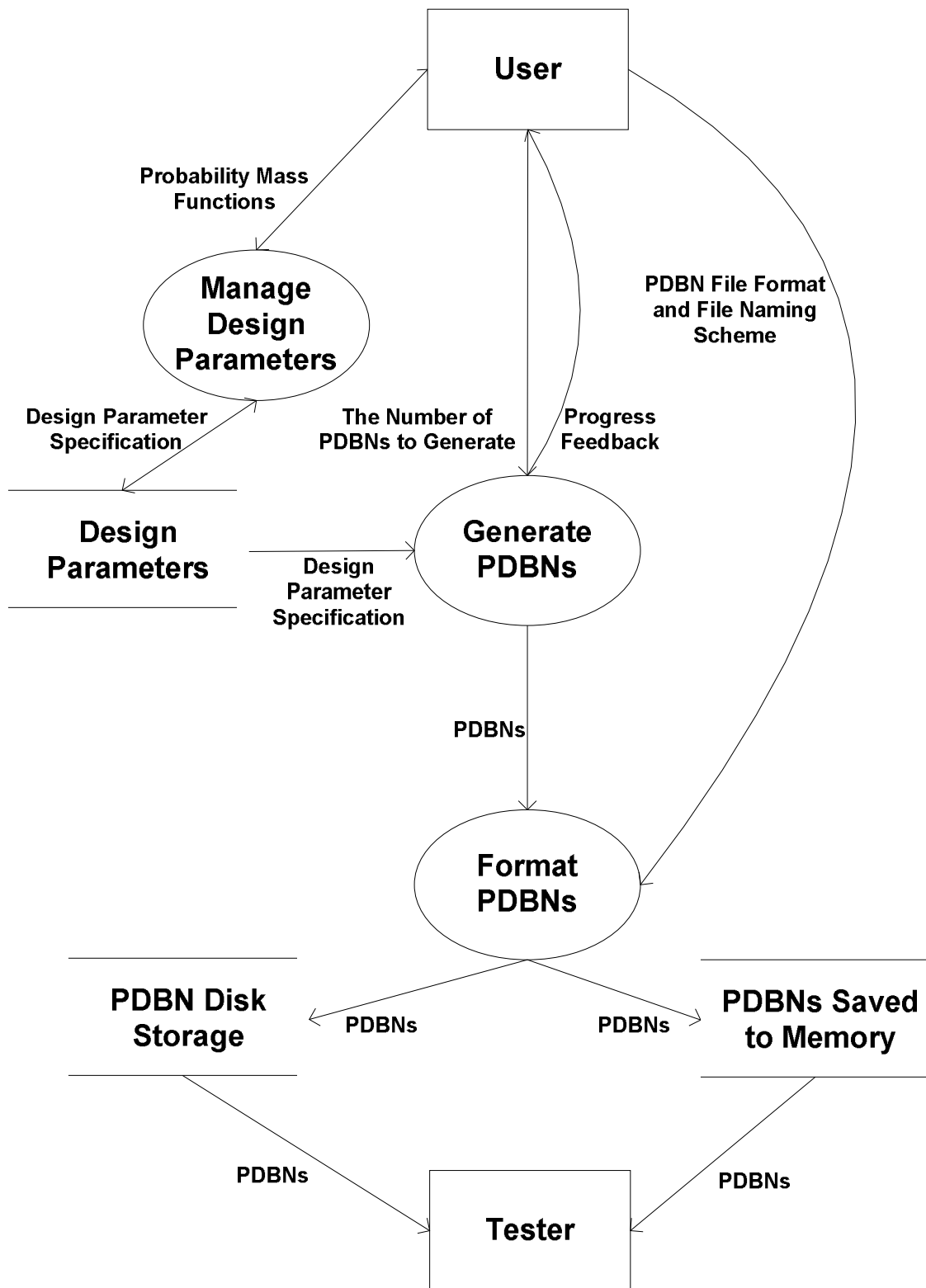


Figure 102: Data Flow Diagram for the Random PDBN Generator

When using the random PDBN generator, the user first defines the taxonomy and design specifications as well as other details such as the number of PDBNs the software will produce. The Random PDBN Generator uses this information to generate a set of PDBNs and change the format of those PDBNs to a standard format that can be easily exported to other applications. These PDBNs are then stored to memory or saved to disk where an automated testing application can access them.

The Random PDBN Generator produces a series of randomly generated PDBNs satisfying a set of constraints as specified by the user. The first set of constraints, the PDBN design parameters, refers to how the PDBN is to be designed. These parameters constrain how the PDBNs generated are to be structured, how each node in the PDBN is to behave, and how the variables represented by the Bayesian network are to represent their states. Table 24 lists these parameters as well as gives a description of each. The Random PDBN Generator takes as input a probability mass function over each of these PDBN design parameters. The PDBN generator includes a data structure for representing these probability mass functions.

Table 24: PDBN Design Parameters to the Random PDBN Generator

	PDBN Design Parameters	Description
1	Number of static nodes	The number of static nodes in the Bayesian network.
2	Number of dynamic transitional nodes	The number of transitional dynamic nodes in the Bayesian network.
3	Number of dynamic non-transitional nodes	The number of non-transitional dynamic nodes in the Bayesian network.
4	Arc saturation for static to static arcs	The percentage of the possible set of arcs between static nodes in the PDBN. 0% represents the minimum number of arcs and 100% represents the maximum number of arcs possible.
5	Arc saturation for static to dynamic arcs	The percentage of the possible set of arcs between static nodes and dynamic nodes in the PDBN. 0% represents the minimum number of arcs and 100% represents the maximum number of arcs possible.
6	Arc saturation for dynamic to dynamic arcs	The percentage of the possible set of arcs between dynamic nodes in the PDBN. 0% represents the minimum number of arcs and 100% represents the maximum number of arcs possible.
7	Arc saturation for transitional nodes between time steps	The percentage of the possible set of arcs between transitional nodes between time steps in the PDBN. 0% represents the minimum number of arcs and 100% represents the maximum number of arcs possible.
8	Probability of dynamic transitional node connecting to self	The probability that a dynamic transitional node connects to itself in the next time step.
9	Average number of states per node	The mean number of states per node.
10	Alpha factor	A node is related to its parents such that $N = \alpha P + \varepsilon$, where N is the random variable of the node and P is the random variable of the parent node. This parameter is the value of the α .
11	Root node mean	The mean of each root node.
12	Noise factor	A node is a linear combination of its parents and a noise factor. This noise factor is a Normal random variable with a mean of zero. This parameter is the variance of that Normal random variable.

Along with the PDBN design parameters, the user has the ability to define various processing parameters that define periphery information necessary to generate the PDBNs. The user also has the ability to define the number of PDBNs to be produced for a given PDBN design specification, the location where these PDBNs will be saved on disk, and the format for which these PDBNs are to be stored to disk. For the purpose of speed and efficiency, the Random PDBN Generator will generate multiple PDBNs at a time using multiple threads. The user has the ability to specify the number of concurrent threads to run at a time to best suit the system the Random PDBN Generator is running on.

To interoperate with the Random PDBN Generator, the automation software of this research study takes the sample set of PDBNs by loading them from file. The files are stored in a file format unique to the Random PDBN Generator. Thus, the automation software of this research study interfaces with these PDBN files by way of a function call to the application program interface of the Random PDBN Generator.

List of References

- Arulampalam, S., Maskell, S., Gordon, N. J., and Clapp, T. (2002). "A Tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking," *IEEE Transactions on Signal Processing*, 50(2), pp. 174-188.
- Boyen, X., Koller, D. (1998). "Exploiting the architecture of dynamic systems." *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, 33 – 42.
- Carpenter, J., Clifford, P., Fearnhead, P. (1999). "An Improved Particle Filter for Non-linear Problems." *IEE Proceedings - Radar, Sonar and Navigation*, 2-7.
- Cooper, G. F. (1990). "The computational complexity of probabilistic inference using Bayesian belief networks." *Artificial Intelligence*, 42, 393-405.
- D'Ambrosio, B., Li, Z. (1994). "Efficient Inference in Bayes Networks As A Combinatorial Optimization Problem." *International Journal of Approximate Reasoning*, 11, 1 – 158.
- Dunn, K., Holt, J., Laskey, K. B., Lyons, B., Takikawa, M., Tung, E. (2002). *Bayesian Networks for Object Classification*.
- IET, Inc. (2002, July). *Test Plan for Evaluating Approximate Inference Algorithms*. Information Extraction and Transport, Inc., Rosslyn, Virginia.
- Jensen, F. (2001). *Bayesian Networks and Decision Graphs*. New York: Springer-Verlag.
- Kleinbaum, D., Kupper, L., Muller, K. (1988). *Applied Regression Analysis and Other Multivariate Methods*. Boston, Mass: PWS-Kent.
- Kuo, F., Sloan, I. (2005, December). "Lifting the Curse of Dimensionality." *Notices of the American Mathematical Society*, 52, 1320 – 1329.
- Laskey, K. (2002, December). Personal Communication

- Lorenz, F. (1987, April). “Teaching about Influence in Simple Regression.” *Teaching Sociology*, 15, 173 – 177.
- Muphy, K., Weiss, Y. (2001). “The Factored Frontier Algorithm for Approximate Inference in DBNs.” *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, 378 – 385.
- Ng, B., Peshkin, L., Pfeffer, A. (2002). “Factored Particles for Scalable Monitoring.” *UAI 2002*, 370 – 377.
- Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo: Morgan Kaufmann.
- Quiddity*. Vers. 5.0. Computer Software. IET, Inc, 2003. <http://www.iet.com/>.
- Takikawa, M., D’Ambrosio, B., Wright, E. (2002). “Real-Time Inference with Large-Scale Temporal Bayes Nets.” *UAI 2002*, 477 – 484.

Curriculum Vitae

Stephen J. Cannon was born on March 18, 1979, in Drexel Hill, Pennsylvania, and is an American citizen. He received his Bachelor of Science in Systems Engineering from George Mason University in 2002. He has 3 years of software engineering experience dealing specifically with artificial intelligence. At MIT Lincoln Laboratory he worked as a summer research student conducting research towards the refinement of object classification algorithms for ballistic missile defense. At Information Extraction and Transport, Inc., Stephen has assisted in the development of a knowledge elicitation environment as well as helped develop and maintain a turn-key system that processes and compiles synthetic test data.