

HONOURS PROJECT REPORT

Weather Forecasting Using Dynamic Bayesian Networks

Report on Dynamic Bayesian Network Learning

Matthew de Kock

Supervised By:

Dr. Hanh Le

Dr. Mark Tadross

Dr. Anet Potgeiter

**Department of Computer Science
University of Cape Town
2008**

I. Abstract

In this paper we present an exploration of the use of Dynamic Bayesian Networks (DBNs) for the purpose of weather forecasting. The work we discuss forms part of a whole project on the subject and concerns the aspects regarding the learning and use of Dynamic Bayesian Networks.

We use Southern Africa as our region of interest and have used data from a collection of weather stations from across the country. Data was available for three variables, maximum and minimum temperature and precipitation. Due to time constraints networks were only constructed and tested for forecasting maximum temperature. However, the algorithms developed apply generally to all variables.

The DBN's are constructed based on already defined intra-slice structures. The networks are built to be representations of the spatial dependencies between the stations in the region of interest. Forecasts are made by adding observed values as evidence to the first time slice of the DBN, then, using inference we can then make forecasts for the subsequent slices.

Machine learning algorithms are used to construct the graphical structure and learn the required parameters. Since the intra-slice structures were already defined by an earlier part of this project the problem of constructing a DBN reduces to feature selection. A simulated annealing approach was chosen due to its ability to avoid local minima. The use of this approach is fully explored within this paper.

Results show some initial promise. However, due to a limited dataset we were unable to perform a full investigation into the potential of DBN's for weather forecast. Initial results are discussed within the context of this limited dataset and a variety of potential avenues for expanding the model are also discussed.

Key Words

Dynamic Bayesian Networks, Simulated Annealing, Machine Learning, Weather Forecast

II. Acknowledgements

The author would like to thank all those who provided assistance and valuable insight during the course of this project. Firstly, the author would like to thank Hanh Le the main project supervisor. Her overview and management helped to keep the project on track and subvert potential problems before they occurred. Secondly, the author would like to thank Mark Tadross of the UCT EGS department. His input to the project on the subject of meteorological issues provided valuable insight to an author inexperienced in the field.

The author would also like to thank Anet Potgieter. Her knowledge of Bayesian Networks and the issues and challenges involved proved an invaluable resource. The author would like to thank Chris Lennard and Chris Jack who provided us with the necessary data. And finally the author would like to thank his two partners for this project, Michael Kent and Pascal Haingura. Their ideas and hard work have served as both motivation and inspiration and without them this project would have improved an unscalable mountain.

III. Table of Contents

I. Abstract.....	2
II. Acknowledgements	3
III. Table of Contents	4
IV. List of Figures	5
1. Introduction	6
2. Background	7
2.1 Weather Forecast	7
2.2 Bayesian Networks	8
2.3 Learning.....	11
2.4 Related Work.....	12
3. Project Description	13
3.1 Causal Modelling	13
3.2 Dynamic Bayesian Network Learning	14
3.3 Visualization System	14
4. Algorithm Design and Implementation	14
4.1 Functionality Required.....	15
4.2 Development Methodology	15
4.3 Development Environment	16
4.4 System Boundaries	19
4.5 System Evolution Description.....	19
4.6 Data Cleaning.....	21
4.7 Data Access.....	22
4.8 Learning Algorithm	22
4.8.1 Simulated Annealing	22
4.8.2 Parameter Learning.....	26
4.9 Prototype Iterations.....	28
5. Additional Theoretical Considerations	29
5.1 Feature Selection	29
5.2 Complexity of Learning	30
5.3 Simulated Annealing	31
6. Evaluation	32
6.1 Testing Methodology	32
6.2 Evaluation Metrics	33
6.3 Computation Measurement.....	35
6.4 Forecast Effectiveness.....	35
6.5 Comparison to Static Network Results	36
6.6 Computational Costs.....	38
6.7 Increasing Degree of Temporal Forecast	40
7. Findings.....	42
7.1 Learning Algorithm	42
7.2 Forecast Ability	44
7.3 Related Work.....	44
7.4 Lessons Learnt	46
8. Conclusions	47

9. Future Work	48
9.1 Multiple Weather Variables	48
9.2 Simulated Annealing Improvements	48
9.3 Optimization	49
9.4 Greater Temporal Dependence	49
9.5 Parameter Learning	49
9.6 Increasing Accuracy of Long Term Forecasts	50
Bibliography	51
Appendix 1: Glossary of Terms	52
Appendix 2: Sample Test Networks.....	52
2.1 Simple Hand Calculated Example Networks	52
2.1.1 Three Node Generic Network.....	52
2.1.2 Four Node Temporal Example	53

IV. List of Figures

Figure 1: Predictive DBN System, showing direction of reasoning	6
Figure 2: Simple Bayesian Network.....	9
Figure 3: Conditional Probability Table (The rows show the outcomes of the variable, the columns the outcomes of any parents)	9
Figure 4: Pseudo Code for SA Algorithm	12
Figure 5: Different Node Types Defined On Temporal Plate	17
Figure 6: Node Types in Unrolled Networks	17
Figure 7: CPT Tables for a Dynamic Bayesian Network Node	18
Figure 8: Search Space Graph (Diameter of 2).....	23
Figure 9: Drop Off Rate for T.....	24
Figure 10: Graph Showing Probability of Accepting a Candidate given the Score Difference for Different Times During the SA Process.....	26
Figure 11: Learning the Parameters of a CPT.....	27
Figure 12: Iteration Schedule	28
Figure 13: Network Accuracies	36
Figure 14: DBN showing where Evidence and Forecasts are made	37
Figure 15: Accuracies of Static and Dynamic Structures	37
Figure 16: Average Time in Milliseconds for Learning Iteration	38
Figure 17: Average Time Spent on Generating a Candidate (Left) Average Time Spent on Scoring (Right)	39
Figure 18: Ratio of Time Spent In Algorithm Sections	39
Figure 19: Iteration Times for Learning a Hundred Node Network	40
Figure 20: Forecast Accuracy for Different Temporal Degrees.....	41
Figure 21: Three Node Generic Network.....	52
Figure 22: Four Node Temporal Network.....	53

1. Introduction

Probabilistic theory and statistics provides a good theoretical base for dealing with uncertainty, a problem typical of weather forecast. However, past statistical strategies have proven to be extremely computationally complex. The advent of Bayesian Networks (BN's) provides a base to apply statistical solutions in a much more efficient way.

The aim of this project was to apply the use of Bayesian Networks to weather forecast. The focus of the project was to create a predictive Dynamic Bayesian Network (DBN) which could be used to forecast weather one day into the future. Automated learning algorithms would be developed and used to construct the DBN's of which several are needed to forecast different weather variables such as temperature and precipitation. Using the large collections of past data available from the Southern African region the models were built and trained automatically.

Using the constructed DBN's, forecasts could then be made by inserting evidence for a number of preceding days and based on this the model would generate a probability distribution for the weather conditions the following days. Based on these probabilities a final forecast can then be made. Figure 1 below shows where evidence and queries would be situated in a simple DBN.

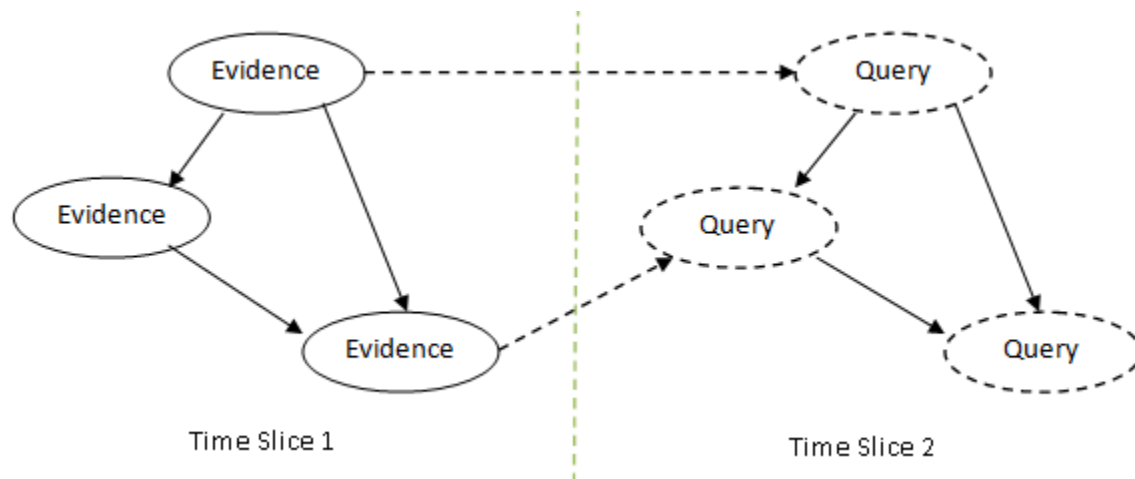


Figure 1: Predictive DBN System, showing direction of reasoning

The construction of the Networks was divided into two key sections, the first concerned with building static models based on inter-station dependencies. The second section, the focus of this report, concentrated on constructing the final DBN based on these static structures and the temporal dependencies between stations.

In this report we explore the effectiveness of the learning algorithm developed as well as the predictive ability of the models these algorithms construct. The learning algorithm used is based on a simulated annealing approach. This is a machine learning technique similar to greedy approaches but with some adaptations which help to prevent the algorithm becoming trapped in local minima. A detailed discussion of the design and development of the algorithm is given as well as an investigation into its effectiveness.

The core investigation of this project will focus on the predictive ability of the created DBN models for weather variables. Several evaluation metrics are available for assessing this ability and will be discussed along with the DBN scores for these metrics. Other aspects of the effectiveness of DBN's will also be investigated within the scope of this project. Key factors such as efficiency and performance will also be investigated.

The final system will also be compared with two existing systems which use Bayesian Networks within the field of weather forecast. The different methodologies, focuses and successes of these projects and our own will be investigated and compared.

This report is structured as follows. First we give a detailed background on topics related to the project. Weather forecasting techniques are described along with short discussion of Bayesian Networks. Next we give a specification of the system and describe individual parts of the project. We then move on to a discussion of the learning algorithms design and implementation. After this we reconsider some of the theoretical complexities and explain how these apply specifically within the domain of the designed algorithm.

We then discuss the techniques used for evaluation and present the results of evaluating our system. These results are discussed within the context of this project and also within the context of similar systems. We end with a discussion of the successes of the project and present ideas for future work.

2. Background

2.1 Weather Forecast

In the past the subjective opinions of meteorologists have made up a substantial part of predictions. Statistical methods have also been extensively used; in particular regression [MURPHY 1984] and the use of Monte Carlo methods have also been common. Statistical techniques provide a framework for measuring the levels of uncertainty that are inevitably involved with forecasting [Murphy 1984].

However, in recent years numerical models have become the focus of weather forecasting efforts. Early pioneers originated the idea that physical atmospheric properties and numerical equations may be used for forecast models. However, these ideas proved infeasible until the advent of computers [LYNCH 2008]. Since their early implementations and as the computational power of modern day systems has ever increased, so too has the complexity and accuracy of these numerical systems. Numerical models are now able to provide accurate forecasts for both medium and long term predictions of both weather and climate.

The ECMWF system uses a numerical system for predictions. It provides forecasts over medium range (10 days) and in addition offers long term predictions of climate and season. The system uses various degrees of range and variables for the different predictions but is based on the physical processes that control the atmosphere [LYNCH 2008].

One drawback associated with the use of numerical models is that they are still relatively constrained by computation feasibility. The complexity and volume of the equations they process means they do not provide enough detail for highly localized predictions on variables that change on more local scales [COFINO 2002]. Methods for down-scaling these predictions have been proposed. Popular methods have been based on the use of analogs, the idea that similar patterns will lead to similar outcomes as may have been seen in the past [COFINO 2002]. More recent methods have used Bayesian Networks as a solution for this problem.

The use of Bayesian networks as a method of representing uncertainty has grown and several weather forecast systems have begun to employ its use. One such system as described by Abramson et al [1996] combines several elements used in the forecasting along with the use of Bayesian Networks. The system which is built as a Bayesian Network also integrates the subjective knowledge of meteorologists to predict severe weather.

2.2 Bayesian Networks

Probabilistic Reasoning

The key advantage provided by statistics is the ability to reason about events given uncertainty. In fact, by assigning potential events a probability of occurrence when we are uncertain if they will occur allows us to easily quantify this uncertainty.

This idea of quantifying uncertainty in probabilities is further developed by Bayes Theorem. Bayesian Networks are based on this idea. Given a particular hypothesis (h) and some evidence (e) Bayes theorem allows us to reason about the likelihood of h given e . The formula is given below.

$$P(h|e) = \frac{P(e|h)P(h)}{P(e)}$$

Given the evidence we can now update our belief in the probability of our hypothesis occurring. We may have several variables available for evidence and they can all influence our belief.

These ideas have been widely used since its advent. However, the areas in which it can be applied in a simple fashion are highly constrained as typical statistical reasoning is an NP-hard problem. However, Bayesian Networks (BN's) provide a way for statistical inference to be used in an automated, efficient way [KORB 2000]. BN's take advantage of the independencies between certain variables within the problem domain to construct a graphical structure, thus reducing the number of dependencies we must consider when performing inference.

Bayesian Networks

A Bayesian Network (BN) models the causal relationships between a set of variables. A BN contains two key aspects. The first is a graphical representation of the dependencies between variables. A directed acyclic graph (DAG) is used to represent this [PEARL 1998]. Each variable is represented by a single node within the graph. Direct causal dependencies are represented by a directed arc from the "causing" node to the node that is affected.

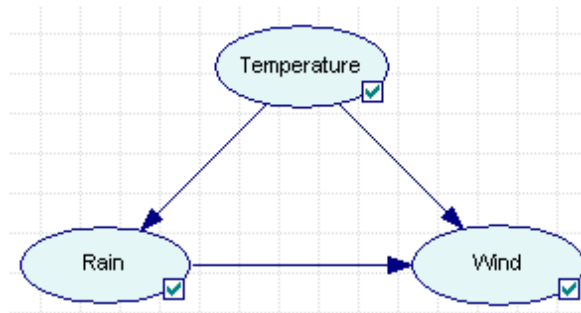


Figure 2: Simple Bayesian Network

The second aspect of the network is a collection of conditional probability tables (CPT's) which represents the probabilities of each state of a node occurring given the states its parents may take. The strengths of relationships represented by directed arcs can be modelled in the probability values stored within the conditional probability tables associated with each node [PEARL 1998]. These values are used to infer the posterior probabilities of each variable given those of its parents.

Temperature	Cold	Cool	Hot
► No	0.5	0.33333333	0.85714286
Yes	0.5	0.66666667	0.14285714

Figure 3: Conditional Probability Table (The rows show the outcomes of the variable, the columns the outcomes of any parents)

Building a model in this way allows us to explicitly show the dependencies and independencies between the variables with a given domain. By using this structure and the assumption of the Markov property, that all dependencies are explicitly modelled [KORB 2000], we can restrict the problem of statistical inference using Bayes Theorem to only use relevant variables as evidence. Due to the structures imposed on the variables the computation required for inference is greatly reduced from the case of performing statistical calculations on the raw data. The representation is compact and efficient [CANO 2004] allowing statistical solutions to be applied to much larger problems than was previously possible.

As variables and dependencies are modelled in a graphical form, it is also easier to visualize, interpret and understand the relationships [COFINO 2002].

Dynamic Bayesian Networks

Dynamic Bayesian Networks (DBN's) as the name seems to imply do not actually have a structure which changes. Rather it is a formalism used to capture temporal dependencies within the field of Bayesian Networks. A Dynamic Bayesian Network is simply an extension of a Bayesian Network which allows us to represent temporal dependencies without the need to create new variables. It contains the same basic DAG structure but adds temporal arcs to capture dependencies between nodes which have some kind of time delay.

The structure defined in a standard Bayesian Network represents the variables at a single moment in time. To model temporal changes using this model would require the definition of new variables to

represent the variables already contained in the model at a point in the future. This quickly becomes an unmanageable solution.

Dynamic Bayesian Networks provide a more simple solution. Rather than defining new variables and new arcs to model these variables it simply repeats the static structure and defines where temporal arcs should exist between these repeated structures or slices. In fact to specify a DBN we now need only specify the static structure and a collection of temporal arcs. We can then repeat the structure as many times as is needed to allow us to model any period of time.

Of course as the number of slices increases the solution again becomes intractable. In typical solutions a sliding window approach is used. We repeat the structure only twice, creating two slices. This now models the change over one unit of time and as time passes we can shift the evidence to reflect the movement of time. This solution can of course be extended to repeat the structure several times depending on how many slices we must model to obtain the desired results.

Inference Methods

As evidence is input into a Bayesian Network inference is performed to obtain new posterior probability distributions for the other nodes in the network. The first algorithms proposed for updating probability were highly limited in scope, applying only to trees or singly connected networks [PEARL 1996].

These original message passing algorithms have been extended and now apply to multiply connected networks in the general case. Despite these advancements the task of updating probabilities for general Bayesian Networks remains an NP-hard problem [Pearl 1996]. However, approximate algorithms for updates do exist and can be used in cases where the estimated complexity is too great [PEARL 1996]. Typically approximate algorithms are based on stochastic simulation and work well even for large networks although the task of inference remains computationally complex [KORB 2000].

The complexity of inference algorithms is largely determined by the number of arcs and the number of states which each node may take up [Abramson 1996]. The number of arcs and states also defines the size of the conditional probability tables. The complexity of variables and the graph structure can in many cases be reduced to allow for more efficient inference. Abramson et al [1996] discuss various methods to reduce the complexity of their BN.

It is also important to note the different types of reasoning that may be performed in a Bayesian Network given the position of evidence and the direction of arcs. Diagnostic reasoning implies that for the node we wish to reason about evidence is available for its children. Predictive reasoning is the opposite of this, evidence is available for the parents and we reason about the states of the children.

This project is concerned with predictive reasoning and within a very specific domain. Given the structure of a DBN all reasoning will be used to determine the values for future time slices. Evidence will be inserted for the first time slice for every node and we will then reason about the states of the nodes in the next time slice. This is shown in Figure 1 on page 6.

2.3 Learning

The construction of Bayesian Networks consists of two elements both of which may be inferred from observational data [KORB 2000]. First the graph structure must be created and then the corresponding conditional probability tables for this structure must be learnt.

Typical approaches for structure learning techniques fall into one of two categories. The first approach is constraint based. This uses statistical techniques to attempt to identify dependencies or independencies between variables in isolation [KORB 2000]. In many cases already identified dependencies may also be used to infer others. A Bayesian Network can then be constructed by analyzing the patterns in these dependencies. In theory a single optimal structure can be defined although this may prove computationally complex. This has been the most widely used approach for learning algorithms to date [KORB 2000]. It provides a relatively simple solution based on statistical theory, giving intuitive results.

The second approach builds a complete structure and then uses scoring metrics to evaluate its performance in comparison to other structures. We can then keep redefining the structure until we have one that provides an effective solution. While being the less used approach experimental literature has shown some favour for the long term use of metric based approaches [KORB 2000].

For general Bayesian Networks the learning of the structure is NP-hard. Approximate algorithms attempt to limit the potential search space through various criteria. In some cases expert knowledge may also be incorporated to help reduce the search space. De Capos and Castellano [2007] explore various methods for building structures with predefined restrictions. The primary use of this is to allow for the incorporation of expert knowledge.

The learning of the conditional probability tables can be performed by mining past data once a suitable structure has been found. Given complete data the process of the learning the CPT values is relatively trivial simply involving counting over a prior distribution [KORB 2000]. When some data is missing the process is more complicated but still feasible.

Simulated Annealing

Simulated annealing (SA) is a global optimization technique which crucially allows the current solution to move to less optimal states based on a probability function, preventing a local optimal from restricting the algorithm.

The algorithm is an analogue of a physical metallurgy technique which involves heating and the cooling materials to increase the size of crystals. The heating process allows the material to enter high energy states which are not optimal and then the slow controlled cooling allows the elements to wander and ultimately settle in a lower energy state than their original.

```

s = s0; e = E(s);
while time < timemax
    sn = neighbour(s);
    en = E(sn);
    if P(e, en, temperature) > rand()
        s = sn; e = en;
    time++;
return s;

```

Figure 4: Pseudo Code for SA Algorithm

SA mimics this process, replacing the material with a current solution and selecting neighbours, similar solutions, as candidates for a new state. A probability function which depends on a global variable T (analogous to temperature) and the energies of the two states is used to determine if the candidate will be adopted. Generally, in the initial stages when T is high, the solution will wander almost randomly through the candidate solutions. T is decreased with time and this means the algorithm will start to prefer an optimal solution more and more as time passes. As time passes the probability of moving to a higher energy state (a less optimal solution) decreases. As T approaches 0, the SA algorithm will reduce to a greedy approach, selecting only optimal features.

The key advantage SA offers over traditional greedy algorithms is that it allows the solution to move to states which are less optimal. This means that it can move away from local optima which are better solutions than their neighbours but not as good as the optimal solution. This ultimately results in a better solution being found.

2.4 Related Work

Here we give a brief overview of other BN based systems that have been developed within the field of meteorology. This is a simple introduction and outline of the aims of these various systems. An in depth discussion of these systems and a comparison with our own system is reserved for section 7.3 Related Work later in this document.

The use of Bayesian Networks in weather forecast has been relatively limited but has begun to grow in recent years. Typical forecast systems have relied either on physical processes or statistics. Few have tried to link the two methodologies. The ability of Bayesian Networks to embody the “causal mechanisms” [PEARL 1998] such as the physical processes underlying weather systems as well as its dependence on statistics for its inference methods could make it the perfect system for bridging this gap [ABRAMSON 1996].

The first system to make use of BN’s was the *Hailfinder* system [ABRAMSON 1996] designed to forecast severe weather in north-eastern Colorado. The *Hailfinder* system was built using expert judgement and

was designed to combine the subjective inputs of expert judgements along with meteorological data to make its forecasts.

Another BN system within the meteorological field was developed by Cofino et al [2000]. Their system was originally developed for spatial downscaling of numerical model forecasts. Since the forecasts generated by numerical models tend to be course grained, typically 50 to 100km apart they developed a BN to spatially downscale these forecasts to give a more local forecast. Their BN maps the spatial dependencies between several weather stations and the grid lines on which the numeric model generates forecasts.

The model created by Cofino et al [2000] was later extended by Cano et al [2004] to be applicable to a variety of meteorological problems including weather forecast.

For the full comparison of these systems to our own please see section 7.3 Related Work of this document.

3. Project Description

This project was proposed by a student, Michael Kent to focus on building a Bayesian Network based system which could be used for weather forecast. The projects main focus is within the computer science field on aspects such as automated learning, artificial intelligence and graphical visualization but also has additional work in the field of meteorology, this being the focus of the computer science aspect, weather prediction.

Dynamic Bayesian Networks offer a potential solution for reintroducing the use of statistics for weather forecasting. This project will use collections of historical data to build Bayesian Networks based on the spatial dependencies between weather stations in the southern African region. Using this model we can use evidence of a past days observed values to update the statistical beliefs we have of what might occur in future days. With these updated statistical beliefs we can then make forecasts.

Due to the project being proposed by a student there were some early problems with defining the exact requirements and scope of the problem. Due to this during the early periods of the project the focus and requirements shifted constantly until the right balance for the overall project and the individual aspects could be found.

In the end the project was divided into three main sections, of which each focused on a different section of the final system. One section focused on the development of visualization techniques to allow forecasts to be displayed in a useful manner. The other two sections focused on the development of the predictive system, each tackling a different aspect of the learning algorithms required. Outlines of these sections are given below.

3.1 Causal Modelling

The first section of the system development was concerned with the identification of dependencies between weather stations for specific types of variables such as temperature. The static Bayesian

Network structure constructed initially is a graphical representation of these dependencies and forms the basis for constructing the temporal Dynamic Bayesian Network. Michael Kent was responsible for this section which had the following goals:

- Construction of Bayesian Network structures for different variable types embodying the inter-station dependencies. This would later form the intra-slice structure for the Dynamic Bayesian Network.
- Exploration of different learning algorithms including Naive Byes Classifier, Greedy Thick Thinning and K2.

The structures created in this part of the project are used in the work covered in this report as the basis on which to develop the Dynamic Bayesian Network.

3.2 Dynamic Bayesian Network Learning

This section focused on developing the final Dynamic Bayesian Network structure based on the developed structures of the preceding section. Given the defined Bayesian Networks constructed in the above section, learning algorithms were used to identify temporal dependencies between stations and construct a DBN based on these dependencies. This section was undertaken by the author and this document details all the design considerations and findings.

3.3 Visualization System

The visualization section of the project aimed to produce a set of techniques to display the generated forecasts within a web based framework. The main technique developed was that of contour plotting which could be used for both the temperature and precipitation variables.

The technique was also adapted to allow for the visualization of these contours as they changed through a short time period. This allows for the display of weather patterns within the developed framework.

4. Algorithm Design and Implementation

The work within this section of the project focused on the development of a general automated learning algorithm to allow for the construction of DBN's based on a given static structure. The final outcome would be a series of developed DBN's for different variables.

The design of the system has one key dependency. It requires a matching series of Defined Bayesian Networks for any DBN's that it will construct. The definition of these initial structures falls under the scope of a different section of the project and forms the only major dependency.

The algorithm also requires a large set of data for the learning algorithm and the assumption is made that all past data is accurate. While errors do possibly exist, within the scope of this project it will be assumed that all input is correct and complete. For a further discussion of this problem see section 4.6 Data Cleaning.

The Bayesian Network will be restricted to modelling only those variables for which past data is available. While several other variables may exist on which those being modelled may depend or influence these will not be included in the model.

4.1 Functionality Required

a. Learning Algorithm

The learning algorithm is the key aspect of design. The algorithm will start with the developed structure for a single time slice and then proceed to learn the temporal dependencies which exist between time slices.

The algorithm will finally output an optimal Dynamic Bayesian Network which can then be used for prediction.

b. Inference

The system will also need a function which will carry out the inference procedure. While standard inference algorithms, available in the software, will be used for this task the system needs to link to them, using the developed Bayesian Network.

c. Forecast and Output

Using the inference algorithms is only one step towards the output. Once inference is complete the forecasts need to be obtained and output into a format that is usable to the visualization side of the project. These outputs were also required during the learning process to allow potential networks to be evaluated.

Forecasts will need to be accessed from the Dynamic Bayesian Network and output into the standardized format.

d. Optimizations

Once the initial structure of the Dynamic Bayesian Network is constructed optimization techniques may be applied to improve the efficiency of the graph for the purposes of inference. Several potential techniques exist for this purpose. A further discussion of these techniques will be commenced at a later cycle of development.

However, they will all function individually, taking as input the default Bayesian Network and updating its structure in some way.

e. Testing Methods

Functions will also be constructed to test the performance and accuracy of the Dynamic Bayesian Network. These methods are needed at various stages. Their use is twofold. The main purpose is to test the effectiveness of the Bayesian Network at forecasting variables through time. The second use will be to test the effectiveness of any optimization techniques used in terms of performance and loss of accuracy.

4.2 Development Methodology

The overall design methodology will follow a rapid prototyping cycle. This means that the working version will be built on in stages, but a working version will be maintained at all times. As new work is completed it will be integrated and tested. This strategy allows for the evolution of the system.

Given that there was some dependency between different sections of the project, using rapid prototyping would allow all concerned to have access to the latest working version of the project and allow them to use this to improve on their own section.

An additional reason for this approach was the complexity of the theory underlying the project. The rapid prototyping approach allows for the development of basic solutions which can be expanded and improved on whilst the background knowledge is grown. This prevented a larger portion of time being spent on research in the initial phase. Time could be spent on development during research, and as new knowledge was gained it could be incorporated.

4.3 Development Environment

To allow for easy development a set of tools were selected which provide a suitable base for developing Bayesian Networks.

Smile and *GeNIe* were selected for this task. *Smile* is a set of platform independent, c++ libraries which provide Bayesian Network support. *GeNIe* is a graphical frontend for *Smile* which runs in a windows environment. Both *Smile* and *GeNIe* were developed by Decision Systems Laboratory.

a. Smile Functionality

The *Smile* libraries provide support for a host of Bayesian Network functions. It provides the basic structures such as networks, arcs and a variety of node types. It also provides key algorithms such as those used for inference of which there are a variety of options including both approximate and exact algorithms. Several learning algorithms are also available for standard Bayesian Networks, although none are currently available in the package for Dynamic Bayesian Networks.

Smile also provides the ability to easily read and write networks to file in a specified XML format. These files are entirely platform independent and may also be easily loaded into the *GeNIe* frontend.

Many of the features of *Smile* were beyond the scope of the project and were not needed. However, the basic functionality and data structures provided a foundation on which our learning algorithms could be developed and tested. All the required functionality for the project was available.

One crucial aspect of *Smile* that should be noted is how some of the features of DBN's are handled. Support for DBN's is a relatively new feature for *Smile* and the techniques used to represent them have some additional features to what is normally associated with DBN's.

Firstly, a node in a DBN may be any of several types as shown in Figure 5: Different Node Types Defined On Temporal Plate and Figure 6: Node Types in Unrolled Networks. Plate nodes are standard Dynamic Bayesian Nodes that would be present in each slice of the network and may contain temporal arcs to themselves or other plate nodes. There are also anchor and terminal nodes which may connect only to

the first or last slice respectively. This means an anchor node may connect to a plate node but will only influence that node in the first time slice. A terminal node may be connected to by a plate node but will only be connected to that node in the last slice of the unrolled network. Finally there are contemporaneous nodes which are not under the influence of time. They may connect with other nodes and will connect to that node for each slice in an unrolled network, but only one occurrence will be found.

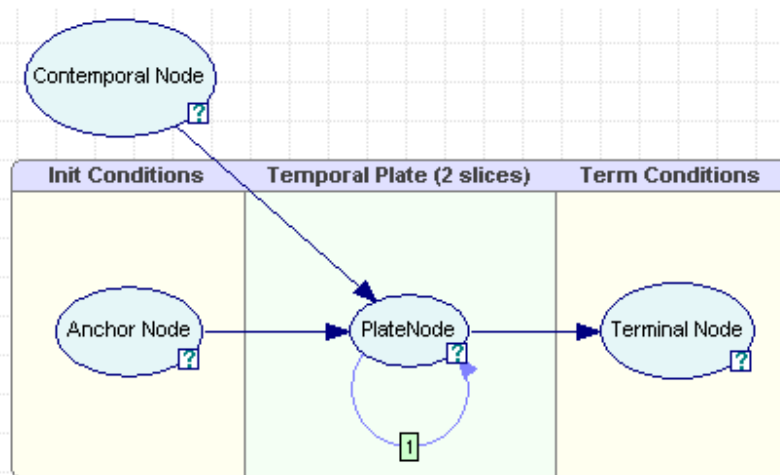


Figure 5: Different Node Types Defined On Temporal Plate

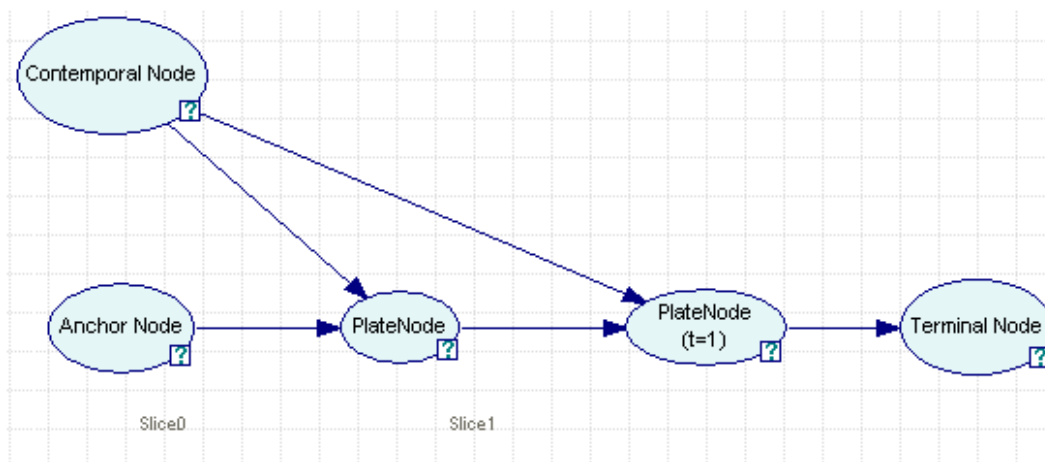


Figure 6: Node Types in Unrolled Networks

Another noteworthy aspect is how the CPT's are stored for DBN's. One table is kept for each temporal degree of arc that connects to a node. This means that a table with a temporal arc connected to it of order one, will store two CPT tables. The first will store the probabilities which are not dependant on time; this will include any normal arcs if they exist, otherwise simply the base probabilities of each state. The second CPT, will also include all degree one temporal parents as well as the normal parents in its CPT.

Rain		No	Yes
	Cold	0.16	0.33
	Cool	0.5	0.45
►	Hot	0.34	0.22

Rain	☐	No			☐	Yes		
(Self) [t-1]		Cold	Cool	Hot		Cold	Cool	Hot
	Cold	0.56	0.33	0.1		0.67	0.32	0.21
	Cool	0.33	0.54	0.41		0.29	0.41	0.42
►	Hot	0.11	0.13	0.49		0.04	0.27	0.37

Figure 7: CPT Tables for a Dynamic Bayesian Network Node

While it is important to note that different node types exist, given the scope of this project and the knowledge that all nodes will be of the same basic temporal type, they will not be used.

b. GeNIe

The *GeNIe* frontend provides a simple way to visualize and build networks. For the most part the *smile* libraries were used for the development of networks and the *GeNIe* frontend was not required. While it allowed for the simple construction of basic test networks, which could be used to test various aspects of the developed algorithms it proved an invaluable tool for viewing created networks in a quick, easy, transparent manner.

The ability to quickly see the created networks and defined CPT's allowed for instant feedback on any changes made to the algorithms. *GeNIe* also allowed for the construction of some simple Bayesian Networks which could be used to test various aspects of the developed system.

c. Support and Technical Issues

Smile and *GeNIe* are both widely used tools and as a result support was readily available. Help documents are available online and for download, and any additional queries may be made in the online forums. Response to any queries was generally quite quick and helpful.

While documentation was generally useful and wide ranging some problems were experienced with regards to the DBN documentation. Due to the relatively new addition of DBN support to *smile* documentation regarding its use was relatively limited. Some documentation was found regarding the methods and classes associated with the newly developed DBN functionality but much of it was incorrect due to some of the refactoring associated with the final integration into the *smile* library.

This did not prove a serious problem in the long term though. While it took a slightly longer period of time to learn how to use the libraries the limited documentation provided at least enough information to give insight into how the libraries might be used, even if not being completely accurate.

Some other problems were experienced with some minor ancillary methods and classes that were documented but did not in fact exist. These missing methods were not crucial for the project though and hence did not cause a problem.

d. Alternatives

Few other libraries exist which have support for DBN's and also contain a GUI. Two such libraries are the *Netica* and *BayesiaLAB* toolkits. While several other toolkits are also available these do not have the benefit of a GUI.

The two systems which contain GUI's are both slow and lack much of the advanced functionality available in the *Smile* library. They are only useful for very small application which can be built by hand [HULST 2006] and so are not suitable for our project. Other libraries without the benefit of a GUI also tend to be slow and cumbersome.

Smile and *GeNIe* provide a good solution with a wide range of features and inference algorithms with the benefit of a GUI and so are the most suitable toolkits for our project.

e. Coding Environment

Since *Smile* using c++ it logically followed that all coding for the project should be done in c++. For this section all development was performed in a Windows XP environment using Visual Studio 2005.

4.4 System Boundaries

As the project is divided into distinct sections and work will be done independently on these three sections, clearly defined boundaries are needed to allow for proper integration and testing.

The three sections are Causal Modelling; Dynamic Network learning and Visualization. Given the large amount of potential data it was identified early on that the use of a common dataset between all three sections was crucial for global testing and development. One general dataset was identified and used for this purpose, though for localized testing other data may have been used in some cases.

The two learning sections were highly dependent on each other as the static structures learnt are then passed to the Dynamic Learning section which then creates a new DBN based on these structures. Using the standard data set simplified this problem and the exchange of BN's could easily be done using *Smile*. Any created network can be saved and loaded from file using *Smile*'s built in methods.

To supplement this each saved network was stored along with the data sets used for testing and training and a header file containing metadata, all saved in standardized formats. This allowed for easy exchange of any networks.

The visualization section was the most independent. Its only requirement was in the form of predicted data sets to display. To allow for ease of development in the early stages it was decided to format any output in the same way as the input. As the output data is attempting to imitate the input as closely as possible in any case this seemed a logical choice and it also allowed for real data to be used for testing and development in the case when no forecasts were available yet from the Bayesian Network.

4.5 System Evolution Description

While the idea was to follow a rapid prototyping methodology and develop each stage based on the knowledge gained in the previous stages a general plan of action was still required. For every iteration of

the prototype key goals and requirements were identified to guide the overall development and ensure that design stayed on track. Below are the outlined plans for each development cycle.

a. Data Cleaning

The first stage of development consists of cleaning the data and making it ready for use. All files were processed to remove irrelevant stations that did not contain worthwhile data, or did not coincide with other data. Files were also processed to ensure they conformed to the agreed upon standard format. Some basic algorithms are also to be designed to allow for simple transparent access to relevant data.

b. Data and Bayesian Network Access

The next stage of development will consist of the design of algorithms to allow both the input of evidence and the output of forecasts for a designated Bayesian network. Input consists of setting evidence in the network according to available past data. Once evidence is inserted, inference can then be performed to obtain forecasts.

Output from the network will require algorithms to read data for forecasts from the developed structure and save them to file, where they can then be used for visualization.

Design at this stage should allow for ease of use later and for the wide range of possible input and output they may be needed at later stages. The amount of data input should be able to vary temporally. This means that the structure should be able to have evidence set for an arbitrary number of time slices. As for input, output should also allow for an arbitrary number of time slices to be used.

c. Initial Learning

The first phase of learning will consist of identifying nodes within the network which should contain temporal arcs to themselves of order one. This basic temporal definition will allow for all basic Dynamic Bayesian Network aspects to be used and tested within a simple environment.

d. Node Generalization

Next the learning algorithms of the previous stage will be generalized to allow for order one temporal arcs to be defined between any nodes within the network. Automated learning algorithms will be fully implemented at this stage allowing for a full Dynamic Bayesian Network to be built containing order one inter-slice arcs.

e. Learning Algorithm Completion

The final stage of learning algorithm design will simply perform basic optimization of the developed algorithm. The algorithm may be further generalized to allow for arcs greater than order one to also be defined depending on performance and time constraints.

f. Testing and Optimization

Before final experiments can be performed, the developed algorithms will be tested for bugs and their performance benchmarked for various size networks. This will allow for the scale of the potential experiments to be determined. At this stage all performance based tests will be carried out.

g. Experiments

The final stage of design will consist of experiments using the network for forecast. Different data sets and different timescales will be used to assess the predictive ability of the network.

4.6 Data Cleaning

a. Raw Data

The data available for the project contained individual station values for three variables. These were maximum and minimum temperature and precipitation. The original data set contained over 3000 stations worth of data for a widely ranging period. The stations were also not operational over the same period but large sets of overlapping periods did exist.

The raw data was stored as a collection of text files, one for each station and data type, as well as a single metadata file for each data type. Each stations data files contained two or three lines of header information along with each day's data value for its period of activity. The header information contained data such as the stations unique identifier, latitude and longitude data, the station name, and the start and end dates of the data values contained in the file.

The metadata file for each data set contained all the header information contained in the individual files but excluded the data values. It also contained information on the number of stations and the global start and end dates for data.

Many of the data files included missing information. This included the occasional missing value as well as extended periods where the station was not in operation and recorded no data.

b. Predicting Missing Data

The decision was made to fill in as many missing data values as possible. In the case of the occasional day missing a value, the surrounding data values may be used to give a good estimate using the expected maximization algorithm. This functions as a converging average and in most cases will provide good estimates of missing data.

No predictions were made for extended periods of missing data. Algorithms will tend to allocate a relatively flat, invariant average to these periods. This does not provide a useful alternative to missing data, as a long period of averaged data values may potentially skew any learning algorithms or later evaluation. Any periods of thirty days or more of missing data were left as is and not used at any stage of implementation or evaluation.

c. Processed Data

To allow for easier access to data the original raw data was processed and cleaned to a degree. Any file containing invalid or faulty data were discarded and where data existed from multiple sources for a single station the source with the most data was chosen. In addition, data was only kept for stations which existed in all the data sets. If data was not available for a station for one or more of the variables it was discarded from the other sets.

In addition to removing unwanted files, the remaining files were also processed to remove the redundant header information and store it solely in the metadata files. Each station was also assigned a number to allow for easier identification and the file names were changed to reflect these station numbers. As some of the original handles were also unsuitable for use in *smile* as they started with integers the new names also took care of this problem.

4.7 Data Access

Due to the large variation in the operational periods of the stations, it was decided to allow algorithms to operate on smaller subsets of the data which would have consistent operational periods. Creation of these sets can easily be performed by accessing the metadata file and identifying which stations operate over the same period.

Given that all algorithms concerned would operate on these subsets, which would be created in large batch processes, it was decided to leave the data in its text file format rather than store it in a database. The creation of the data base would create unwanted overhead, while not providing much benefit given that only a few queries would ever be made throughout the course of the project. In fact, given that a common dataset was identified and used for all testing only one query was ever made.

A simple program was designed to create the subsets of data for specified periods, by first accessing the metadata file to find which stations were operational during that period and then to access the individual files to retrieve the data. The resulting subset could be split between training data and test data and was saved to file, in addition a header file containing the basic information for the set was also created.

Given that this kind of access would occur frequently and that the subsets are relatively small this approach is more than adequate for requirements. The use of a database was considered but because of the additional overhead required and the infrequent queries it was not deemed to be beneficial.

4.8 Learning Algorithm

The learning algorithm serves to build the final Dynamic Bayesian Network structure that will be used for forecasting. The algorithm must take in the defined intra-slice structure in its final form and then learn the temporal arcs for the DBN.

Given that the intra-slice structure is defined, the problem of defining the temporal arcs reduces to feature selection [MURPHY 2002]. Various solutions exist to solve this kind of problem, all giving approximate solutions as the search space for potential BN's is super exponential.

4.8.1 Simulated Annealing

The SA algorithm requires the definition of several key elements. Each of these elements can be defined individually and used by the SA algorithm as a black box. This provides a key advantage given the rapid prototyping methodology adopted for this project. Each element of the SA algorithm can be defined independently and then changed and updated at each stage of the development without the need to change other parts of the algorithm. Each of these elements is explained below and the implementation within our solution is given.

a. Neighbour Selection

Selecting neighbours has a wide ranging effect of the algorithm and subtle changes to this process can greatly increase the effectiveness on the algorithm. First consider the search space as a graph, each node representing a possible state and each arc a transition between the two states.

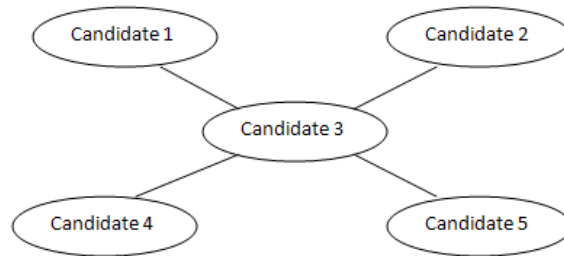


Figure 8: Search Space Graph (Diameter of 2)

An important characteristic of the neighbour selection technique is that it minimizes the distance in this graph between any solution and other possible solution which may be the optimal solution. The state space graph should be sufficiently small in diameter to allow the selection algorithm to easily move between possible candidates.

The selection algorithm can also optimize the graph by minimizing collections of nearby states with local minima. This means that if a large number of connected nodes all have similar low energy, forming a local optimal the state can get stuck wandering in this collection for long periods rather than moving away from the local optimal.

Another feature of the algorithm is that the candidates it generates should be similar to their predecessors. The algorithm should attempt to prevent very good and very bad candidates, which can cause the state to wander almost randomly.

Implementation

To select neighbours for our Dynamic Bayesian Network a simple method was used. Two nodes were simply selected and if no arc existed one was added. Alternatively if an arc did exist it was removed. This simple approach meets many of the criteria discussed above.

Each candidate is only slightly different to its predecessor, differing by only one arc. This should make only a small difference to the overall networks predictive ability, especially given the large size of some of the networks. In fact, even for very small networks of only a few nodes very little difference is seen in the predictive ability.

Also since neighbouring candidates differ by only a single arc, the solution is unlikely to spend time moving between collections of local optima's grouped together. However, since the diameter of the search space graph is still large, $O(n^2)$, the problem of moving from one candidate to another potential optimum still exists.

In addition, the candidate generation is relatively efficient. All that is required when adding or removing an arc is to relearn a single probability table.

b. Temperature (T)

The temperature value is a key global parameter. It controls how the algorithm shifts over time to favour the more optimal solutions and reduce the “random wandering” of the SA algorithm. The T value used in the algorithm was defined as a function of time and always took on a value between one and zero. Its value at any stage is defined by the annealing schedule described below.

c. Annealing Schedule

The global parameter T must change over time and a suitable schedule must be implemented. If T drops too fast, we once again have the problem of getting stuck at local optima's, but if it drops too slowly, the solution will keep wandering randomly, which does not provide a benefit either.

Implementation

Given that the learning process is slow, an exponential drop off rate was selected for the annealing schedule. This means that initially T drops off very rapidly, reducing the time spent in wandering the solutions. As T begins to approach 0 the drop off rate decreases and more time is spent during the more greedy section of the algorithm.

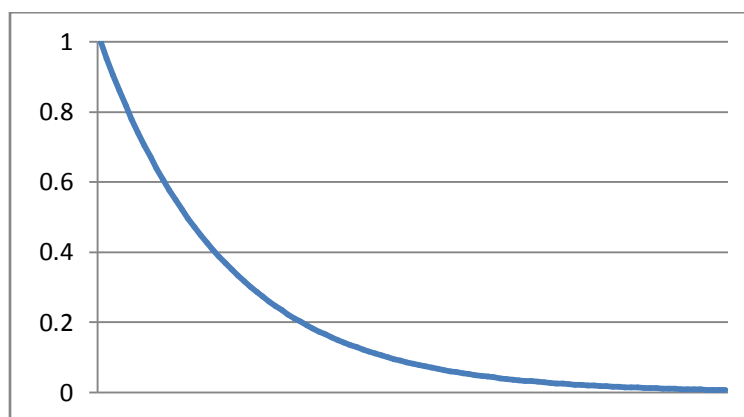


Figure 9: Drop Off Rate for T

d. Probability Function

This function serves to choose whether the current state will move to a particular candidate state. It is highly dependent on the parameter T, which affects the probability of moving to higher energy states. Given T, together with the energy of the current state and the energy of the candidate, this function must return a probability giving the likelihood of moving to a selected candidate.

In our case there is also an additional consideration to be made when evaluating whether to adopt a candidate solution. We must also consider the additional complexity of the candidate. Since for Dynamic Bayesian Networks the theoretically optimal graph structure in most cases is the fully connected graph, which is not the real optimal due to the computational complexity. Due to this when adding arcs, the function must receive a penalty and when removing an arc it should receive a slight bonus.

Implementation

The normal distribution provides a good function for generating the probability of accepting a candidate. The function for the normal distribution has a peak around a selected mean which in our case we replace with a selected optimal improvement. As the scores given to the function move further from this optimal improvement the probability of acceptance decreases at a rate determined by the variance which we replace with a parameter dependent on our current T value. Below is the general formula for the normal distribution.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

μ and σ represent the mean and the variance of the distribution respectively.

The normal distribution reflects the base on which we create our function. Below we show the full adapted function which we use. We then describe each parameter.

$$P(\text{Accept Candidate}) = \frac{2-T}{2} * 2 \left(\frac{1}{(0.025\sqrt{2\pi})} \right) e^{-\frac{((\text{Candidate PA} - \text{Current PA}) - 0.025)^2}{2T^2}}$$

As we have mentioned, selecting an optimal improvement of the score will determine which candidates are favoured. Large differences are not favourable as the candidates may become too random. Given this our optimal improvement will be positive (to ensure improvement) and low, close to zero. For the final solution a value of 0.025 was selected. This reflects an improvement of 2.5%. This can be seen in our equation replacing the μ variable.

The other parameter we change is the variance (σ) of the normal distribution. This effects how rapidly our probabilities will drop off as they begin to differ from our selected optimal. Considering that early on we would like solutions to exhibit slightly more random behaviour and as time passes the solution imitates a greedy algorithm more and more closely, it was decided to use a value dependant on the T value for this parameter. Since T is always between one and zero it already provides us with a good value. We can simply substitute our current T value into the equation.

A final additional factor is also used to increase the spread. The first factor seen in the equation is also dependent on T and serves the purpose of reducing the favour for optimal solutions in the early stages. This serves to increase the spread of the function in the early stages. Again as time passed and T approached zero the effect of this factor became negligible which allowed optimal improvements to be favoured.

As can be seen in the graph below, as time passes the selection window for candidates gets narrower, increasing favouring the optimal improvement. The final factor, seen second in the equation, is used to keep the scores between one and zero (to reflect probabilities). It simply divides the scores by the theoretical maximum score for that step.

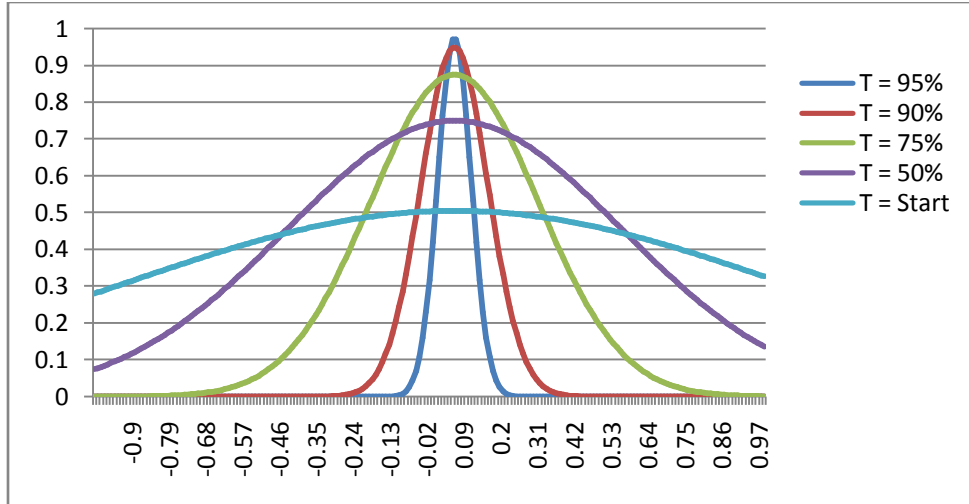


Figure 10: Graph Showing Probability of Accepting a Candidate given the Score Difference for Different Times During the SA Process

The final score was then multiplied by either 0.95 if an arc was added or 1.05 if an arc was removed to reflect penalties/rewards for changes in complexity.

e. Energy

Each state of the solution has an associated energy which depends on its effectiveness as a solution. A method must exist to calculate the energy of each state.

Implementation

The energy of each state is determined by the predictive ability of the network associated with that state. Several metrics exist which can determine the effectiveness of a given network compared to a given dataset and these are discussed later in the paper under the evaluation section.

However, given that this scoring metric will be repeated numerous times during the learning process, making efficiency important the simplest measure, predictive accuracy, was selected to determine each candidate's energy. For a full explanation of how the predictive accuracy is determined please see section 6.2 Evaluation Metrics.

4.8.2 Parameter Learning

Learning of the conditional probability tables forms a crucial part of the learning algorithm. Given the structure of the network the parameters for each node must be learnt from the data. During the learning process, each time the structure is changed the parameters must also be re-learnt to allow inference to be performed and thus the changes evaluated.

Given that the datasets used for learning contained only complete data, the parameter learning can follow a relatively simple approach. Given the number of states a node and its parents can take in combination (giving the full CPT) we can simply sum the frequency of each combination over a given prior distribution. For most cases this prior distribution can simply take the form of a relatively low valued uniform distribution and typically for this project a distribution of uniform 0's or 1's was used.

For instance, if a child has two states and a parent with three states, there are six possible combinations of their states. A count is then performed for each combination and added to the prior distribution to find its probability relative to the other counts. So we would count how many times state 1 of the child and state 1 of the parent occur together within the data. We then count how many times state 2 of the child occurs together with state 1 of the parent. Then by dividing these counts by the total of the two we can get the probabilities for the first column of the CPT, as shown below.

Parent State	1		2		3	
Child State	1	2	1	2	1	2
Count	3	7				



Parent	State1		State2		State3
State1	0.3		0		0
State2	0.7		0		0

Figure 11: Learning the Parameters of a CPT

This would be the probability table for the above example after counting for the first two combinations of states. This table would be produced by counts of 3 and 7 respectively using a uniform prior of 0's. This is performed for all the values in the CPT.

Given that *smile* stores both the references to parents and the CPT's in a different manner for the normal and temporal case, different methods had to be designed to handle either case. The difference between the two methods was only slight. The retrieval of the parent nodes for the temporal case simply requires access to two sets of parent references rather than one and a flag had to be set to note whether a particular parent was temporal or not to allow the correct data element to be accessed when counting.

Each nodes CPT is independent of the others and will only change when the number of parents changes. Given this we can restrict the update procedures at each stage of the learning to only update those CPT's which have been altered. This simple restriction improves the efficiency of the algorithm during the structure learning process.

The algorithm however, does not take into account parameter dependencies and as a result can be slow compared to some other methods which take into account these dependencies [KORB 2000]. However, the methods simplicity means it is still widely used and very effective.

One potential problem that may be experienced during parameter learning concerns the volume of data. As the size of the CPT tables grows exponentially with every added parent the number of parameters that we are required to count over can easily grow larger than the volume of data we have available.

For instance, given that we used five states for every node, a node with no parents has five parameters to learn. Given that our training set contains over six hundred values this is easily accomplished. Adding a single parent still means there are only twenty five parameters to learn, still easy. But if the number of

parents hits five we are suddenly required to learn three thousand parameters. Given the now relatively small size of our training set an effective CPT is difficult to produce.

4.9 Prototype Iterations

Development of the algorithms used to generate the networks was done in stages as a prototype was developed and improved on. Initial work focused on foundational areas which would be needed to run and test the learning algorithms. Due to some of the dependencies involved in the project, work on the learning algorithms could not begin immediately and so this foundational work was the focus of initial work.

After this foundational work was completed, the design of the learning algorithm was initiated. First the basic skeleton of the algorithm was developed and then the individual elements were expanded on. Figure 12: Iteration Schedule shows the basic focus of each iteration. Each step is explained below.

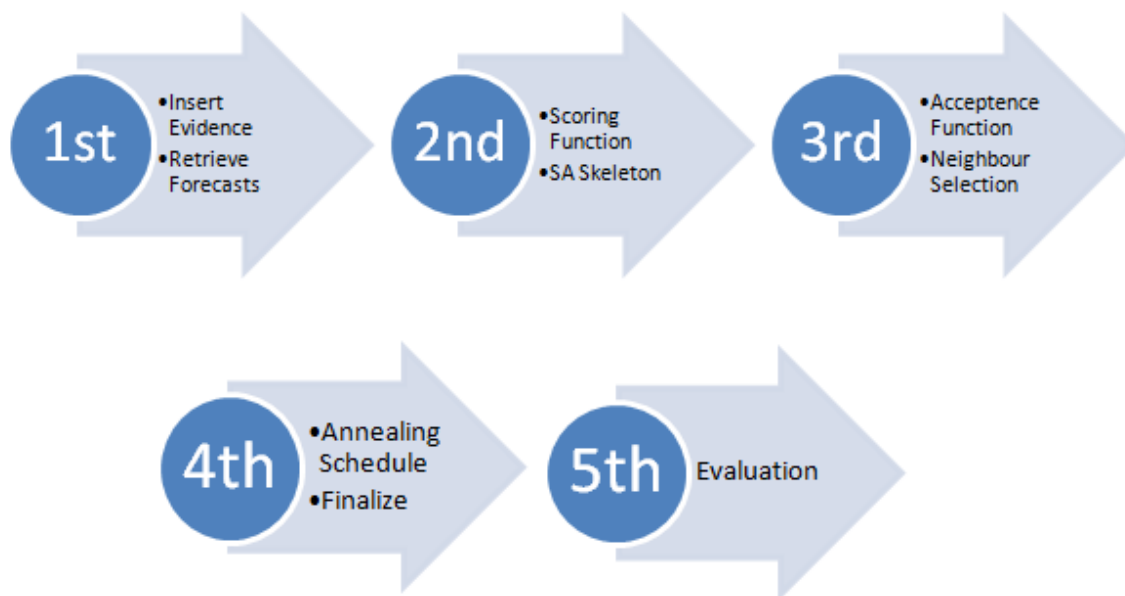


Figure 12: Iteration Schedule

1st Iteration

Given that some initial data was required from other sections of the project, early development of the learning algorithm was delayed slightly. With this in mind the first stage of development focused rather on tools that would be needed later, such as evaluation techniques.

The first stage developed a system to insert evidence from a file into a network, and then retrieve forecasts made. More details of this algorithm are given in the evaluation section.

2nd Iteration

This was the first stage of the development of the learning algorithm. This first iteration focused on developing the skeleton structure of the SA algorithm, which would allow for the expansion of individual elements at each future stage.

This skeleton would consist of many of the elements required for the SA algorithm but in a basic initial form. The energy scoring function was developed in full and used as the sole criteria for selecting new networks, since the probability function was not yet developed.

Candidate networks were selected in a simplistic form. The algorithm simply iterated through all the nodes in the network and a candidate was chosen by adding a degree one temporal arc from each node to itself. If the arc improved the score it was kept, otherwise the next node was tried.

Key SA features not present at this stage are the T value, the full probability function which allows candidates of lesser score to be used and a more sophisticated candidate selection procedure.

3rd Iteration

The second iteration began to build the individual parts of the SA algorithm. First the neighbour selection algorithm was expanded from the initial version to the final general version which could select any nodes to place arcs between.

A preliminary version of the candidate acceptance function was also developed. The initial version built the basic elements, retrieving the score difference, the dependence on T and the punishment for complexity. These basic blocks could then be tinkered with until a final solution (the normal distribution function) was selected.

4th Iteration

In the final iteration the last part of the SA algorithms were implemented. The final exponential drop off annealing schedule was implemented.

The candidate acceptance function was also built into its final form through some simple experimentation and trial and error.

5th Iteration

Once the learning algorithms were complete, the final code required for evaluation was implemented. This simply involved adding variables to keep track of the time taken in various aspects of the process as well as to record the scores of candidate networks. All the saved variables were simply written to file from where they could easily be loaded into a spreadsheet package for analysis.

5. Additional Theoretical Considerations

5.1 Feature Selection

While the overall task of the project is to develop algorithms to build Dynamic Bayesian Networks to be used for weather forecast the task is divided into two key areas. Initially the static structure must be defined representing the inter-station dependencies. Next and the focus of this section of the project and report, is to define the temporal arcs that make up the Dynamic part of the network.

Given that the intra-slice structure has already been defined, and only the inter-slice temporal arcs now need to be learnt the learning problem can be reduced to a feature selection problem [MURPHY 2002].

The problem is simplified from the general network learning case as the direction of all arcs is known. This is because temporal arcs must always move forward. The child node must always be in an antecedent slice from its parent.

Due to this restriction during the learning process it is not necessary to check for the creation of cycles within the graph. As no arcs can have a child in a precedent time slice it is impossible for cycles to form.

Given this simple restriction to the problem the only task left is to select which variables are temporally related to each other and provide us with a benefit to our forecasts. This is a feature selection problem.

5.2 Complexity of Learning

For the general case the learning of Bayesian Networks is computationally complex. The search space for potential networks is super exponential making exhaustive searches impossible for all but the smallest cases. Because of this problem any learning algorithms must attempt to limit the search space.

The main restriction considered by our learning algorithm is the reduction to feature selection. The intra-slice structure is taken to be known and optimal and so we need only learn the potential temporal arcs. We can assume these intra-slice structures are optimal since they are the focus of the casual modelling section of the project.

There are also additional external factors which we must consider that have a great effect on the complexity of the learning process. The greatest of these in the case of this project is the number of states which each variable can take on. Since our BN's use discrete distributions we must quantize our data into a number of defined states.

The larger the number of states we allow the more potentially useful the forecasts become since they will provide a closer representation of the real state. Given extra states we can define a forecast closer to a particular value. For instance if we have five states for temperature each covering five degrees we can only be accurate to within five degrees. If we create more states each representing two degrees now we can see that forecasts will be closer and more accurate.

Conversely learning and inference both increase in complexity and accuracy will be likely to drop since there are a greater number of states which forecasts may take. If there are only five states, at worst picking randomly we have a 20% chance of a correct forecast. If we increase the number of states this chance begins to drop.

Those external considerations aside it is on the learning where the key effect is noticed. As the number of states increases so to does the size of the CPT's. Take a node with two parents, with five states its CPT has 125 parameters. With ten states this number jumps to 1000. Clearly this will mean the learning process takes longer and given our concerns over the volume of data used for training this consideration is even greater.

Just as the size of the CPT effects the complexity of parameter learning it also affects the complexity of inference. Since inference makes up a significant part of the learning process as well the effects are twofold. Given these considerations it was decided to allow each variables to assume five possible states

defined based on the data itself. Given the concerns of this project this seemed a reasonable number allowing the volume of data to sufficiently be used for parameter learning and still giving good insight into the ability and usefulness of the forecasts generated.

5.3 Simulated Annealing

There are two key theoretical issues associated with SA that we consider here. The first is the benefit it provides by potentially allowing weaker candidates to be picked to avoid local minima. We discuss this issue first. Secondly we also look at the complexities of the search graph associated with the particular implementation of SA we have used.

a. Avoiding Local Minima

This is the key benefit of simulated annealing. Typical greedy algorithms are associated with the problem of becoming trapped in local minima which provide immediately beneficial solutions but are not as good as other potential solutions which may be found. SA avoids this problem by allowing the solution to move to higher energy states by accepting candidates which are less optimal.

The degree to which a particular implementation will allow this to occur can be controlled by the selection of the annealing schedule and the candidate scoring function. Typically the candidates selected in the early stages will allow for a greater number of weaker candidates to be potentially accepted and by the final stages the selection will follow an almost greedy approach.

When using an SA approach within the field of BN learning this benefit also applies in a slightly different context. Strictly speaking whenever we add an arc the new network will at worst be as good as the previous because the CPT's can be constructed to simply represent no relationship even if we place an arc there.

Where we see this benefit is mostly when we consider arc removal. When we remove an arc from the network, the new candidate can indeed be less accurate. However, the computational complexity of the network also drops which is to our benefit. Using SA we can allow these candidates to be accepted within a simple framework and thereby construct networks which are not only accurate but also computationally efficient. This is a crucial idea given the scale a real world application within this field would take.

b. Search Graph Complexities

We have already discussed the potential effects of the candidate selection algorithm on the size of the search graph. A poor candidate generation technique can have a large detrimental effect on the efficiency of the SA algorithm.

Our candidate generation technique has resulted in a search graph of a diameter equal to the number of nodes in the network squared. Since we select one arc at a time each of the nodes in our search graph will differ by just this one arc. Since we can have an arc between any two nodes including themselves to move from any one candidate in the search graph to another we would have to change at most every arc in the current solution. This means that for every node and every other possible node we would

have to add an arc if there is not one already or remove the existing arc. This gives us our $O(n^2)$ complexity where n is the number of stations we are considering.

An $O(n^2)$ algorithm is still acceptable given the constrained size of our networks and given the simplicity of our candidate generation this complexity is acceptable. However, for a larger application this technique may need to be revised.

6. Evaluation

The goal of this project was to create a Dynamic Bayesian Network to forecast weather. To assess how effective our generated networks have proved to be a series of experiments were carried out to assess various aspects of their results. Key factors to be assessed are the accuracy of predictions, the computational cost of learning and forecast generation as well as differences between generated networks and how they are used. Below is a description of each experiment, outlining the goals, the methods used and the results obtained.

Each experiment was carried out on several different networks of different size. The number of nodes in each network used was ten, twenty five, fifty and one hundred. The use of different sized networks will give us some idea into the overall scalability of the project given that a full scale network could not be created due to time and data constraints.

The networks used only the data for max temperature. Again due to time constraints networks could not be created for all the variables. However, the forecast ability should not change significantly between different variables so again we can gain insight into the potential of DBN's even with only a single variable being used.

6.1 Testing Methodology

The testing for this project was performed in a semi-structured way. Each class and method was debugged as it was written using ad-hoc examples and basic output. This testing was unstructured and informal and was used simply to remove syntax and basic logic errors present within the code.

More formally, for some larger more significant methods black box test cases were developed to test for errors. Depending on the function of the method a small sample problem was developed and the class used to solve this. These tests are performed with simple debugging statements on and all data output.

The algorithm was also repeated tested on small networks to ensure that errors did not creep in unexpectedly.

Prototype Iteration Evaluation

Each iteration of the learning algorithm was tested with a set of networks. These networks provided a range of different problems such as the size of the network, the number of arcs and the sizes of CPT tables. Several small networks were calculated by hand to test the probability learning capabilities. As the size of networks increased this kind of test becomes increasingly difficult.

Testing was initially performed using these small controlled examples to help locate and correct any simple logic errors. The larger test cases were then used to stress the system and locate any other errors.

6.2 Evaluation Metrics

When evaluating probabilistic forecast models there are two main aspects which we must seek to maximize [KORB 2000]. These are:

- **Domain Knowledge:** This is the frequency with which the model can correctly predict the correct outcome [KORB 2000].
- **Calibration:** This is the ability of the model to predict its probability distribution close to the actual frequency or distribution of the real event [KORB 2000].

These two criteria can be maximized together and are not independent. One can clearly see that if the second property is maximized, that is we predict events with a probability distribution exactly matching that of the true distribution then there can be no more domain knowledge to be gained [KORB 2000].

There are three metrics which could all provide insight into the success of our developed models. These are described below.

a. Predictive Accuracy

Predictive accuracy (PA) is a simplistic but widely used and effective metric for scoring predictive systems. It is simply a measure of how often the system is able to predict the correct occurrence of selected variables.

Given the distributions calculated by the model we can simply take the event with the highest probability as being the models prediction. Correct predictions will increase the score while incorrect forecasts do not.

This measure is simple to calculate and gives very good insight into how effective the model's domain knowledge is. However, it does not take into account the models calibration. The distribution given by a model may be relatively uniform, meaning that selecting one event as the outcome over another is relatively arbitrary.

For instance consider if a model gives a probability that a certain type of mushroom is edible is 0.51 and that the probability that it is poisonous is thus 0.49. This means that if the mushroom is in fact edible the model will score a 100% given this single test case. However, this is not a fully effective measure of the models ability. The confidence of any predictions is entirely ignored [KORB 2000]. Given the initial probabilities we might rightly hesitate to consume the mushroom given that it has a 0.49 probability of being poisonous.

While providing a good initial idea of a model's ability, other methods might provide better insight into the models calibration as well as its domain knowledge.

b. Bayesian Information Reward

This metric is an extension of I.J. Good's information reward (IR) metric. The IR metric assumes a uniform prior probability distribution, meaning that models with variables containing unbalanced distribution may be incorrectly evaluated. The second deficiency of the IR metric is that it applies only to binomial predictions; that is systems with only two outcomes.

The Bayesian IR metric extends Good's method by taking the prior distribution into account and generalizing it to multinomial distributions [KORB 200].

The Bayesian IR (IR_B) is given by:

$$IR_B = \frac{\sum_i I_i}{n}$$

Where n is given by the number of test cases and for the true case I_i is given by (1) below and otherwise is given by (2).

$$(1). I_i = \log \frac{p'_i}{p_i} \quad (2). I_i = \log \frac{1-p'_i}{1-p_i}$$

p'_i is the probability given by the model that the variable takes on a certain value and p_i is the prior probability of the variable taking on that value [KORB 2000]. This method is closely related to the Kullback-Liebler Divergence (described below) but replaces the reference to the true probability with the prior probability distribution [KORB 2000].

Two key aspects to note are that the score is weighted by the number of occurrences within the test set. Secondly the ratio between the forecast and the prior is inverted. This reflects the fact that the model is a reward for diverging from the given prior (i.e. shows the ability to account for evidence and update its distribution) so long as this gives a benefit to PA.

So we can see that if a prediction is made that is exactly equal to the prior distribution the score will be zero. This means that ignorance is never rewarded. Taking into account the relative probabilities of any predictions now means that we account for property two as well. As opposed to PA this metric also gives us an insight into the calibration of the model as well as its domain knowledge.

c. Kullback-Leibler Divergence

The KL divergence is a measure of how closely the posterior predicted distribution of the model matches the true distribution. This is an idealized metric which requires knowledge of the true distribution of any variables predicted by the model [KORB 200]. If the KL divergence is minimized it implies that the model is perfectly calibrated so clearly both the above properties are measured by this metric. The KL divergence can be measured using the following equation:

$$KL(p, q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}$$

Where $p(x)$ is the probability given by the model and $q(x)$ is the probability given by the real distribution.

However, given the requirement that the true distribution must be known which is untrue of all real world models the measure is not useful to most application. Were the real distribution known we would have no need to collect data and build the models to be assessed by the KL divergence [KORB 2000].

6.3 Computation Measurement

Aside from the obvious concern of the accuracy of our predictions it is also important to measure the computational costs of any of the networks we use for forecast. The feasibility of using Dynamic Bayesian Networks for weather forecast is as much dependent on the computational expense as on the possible accuracy.

All measurements were performed on the same machine to ensure consistency throughout the acquired results. The computer used was a laptop with a 2.8 GHz Pentium 4 processor and 448 MBs of RAM using a windows XP operating system. Although the power of the computer used for testing was low, given that tests were also performed on small sized networks good insight may still be found into the potential scalability of future solutions.

All measurements were performed using the standard windows c++ function `GetTickCount()`. This function provides relatively good accuracy for the required purpose, giving measurements in milliseconds accurate to within fifteen milliseconds. The only potential problem is that very short periods of fewer than fifteen milliseconds may not be captured accuracy.

However, given that our main concern is on a much larger scale and any measurements of this small nature would be insignificant within the broader view of the system, the measurements provided by this function are more than adequate.

6.4 Forecast Effectiveness

This is the primary assessment criteria for the project. Whatever the costs and abilities of the learning algorithms it is how effective the generated networks are at forecasting max temperature that will decide the success of the project.

As for learning our primary measure of a networks performance will be its predictive accuracy. A full discussion of how this measure is calculated is given earlier on the paper.

Given each Dynamic Bayesian Network we have learnt, one for each size, we will calculate the predictive accuracy using a test set of 67 days data. This data is for the period directly after the data from the training set. The results are shown in Figure 13.

We can see from this graph that there is a general increase in the forecast ability as the model size increases. The two smaller models show a forecast accuracy of around 53% while the two larger models are over 2% better than this. While on the surface this difference seems only small if we consider that in a real world application the network may contain potentially many thousands of nodes this increase may grow to a much more significant amount.

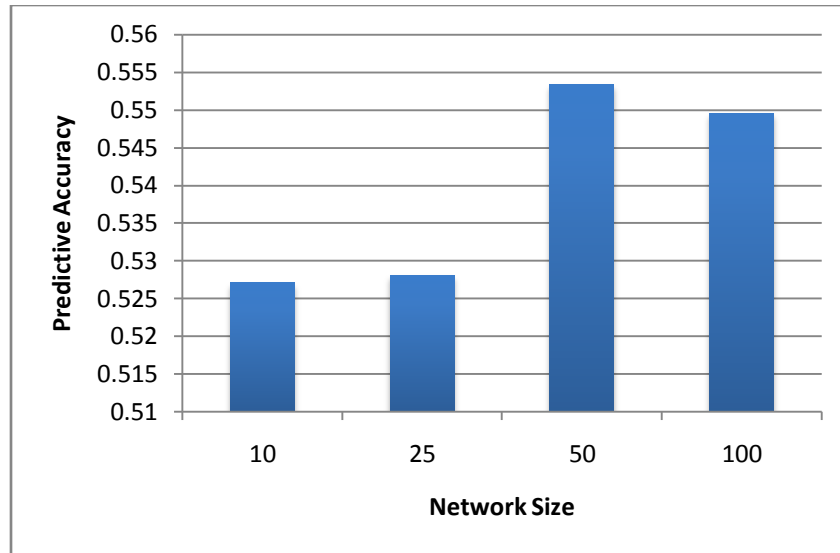


Figure 13: Network Accuracies

As a larger network contains more stations it seems likely that the larger the network we use the more likely we are to find good causal dependencies between two of the stations. The more of these dependencies we can find the more accurate our forecasts will become. So we can hypothesize that a larger network will provide a benefit the accuracy of forecasts when compared to a smaller network.

What is slightly unexpected is the drop in accuracy when the model is increased in size from 50 to 100 nodes. However, it is most likely that this is simply an anomaly of the learning procedure. Since the algorithm is non-deterministic it will not create the same network each time. Because of this it might simply have built a particularly poor 100 node network or alternatively a good 50 node network.

Due to the time constraints we were unable to repeat the learning procedure several times for the larger networks and so it is possible that the algorithm simply did not perform as well in this case. The difference is only minor, less than 0.4% but in typical cases we would expect to find an increase in accuracy. Given more time the learning process could be repeated several times. This would allow us to get an idea of the average ability of the created networks and establish a confidence interval for the constructed accuracy.

6.5 Comparison to Static Network Results

We have seen now how accurate our DBN's forecasts have been but it is important to note whether or not they provide a benefit over a normal Bayesian Network. Within the context of this project the primary benefit of using DBN's means that we can add an entire days observed values as evidence into the first slice and then make a forecast for the next slice based on this evidence.

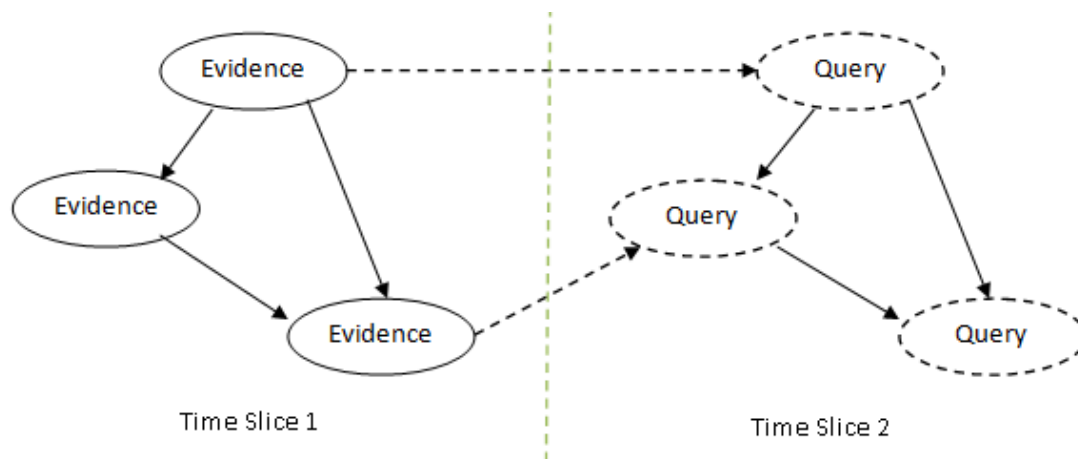


Figure 14: DBN showing where Evidence and Forecasts are made

Considering this the forecast accuracy of a DBN should provide a huge benefit over the normal network on which it is based because essentially without temporal arcs we cannot add the previous day's evidence and so a static network is completely ignorant. The only evidence on which it can base its forecasts is the probability distributions it contains, no evidence may be added.

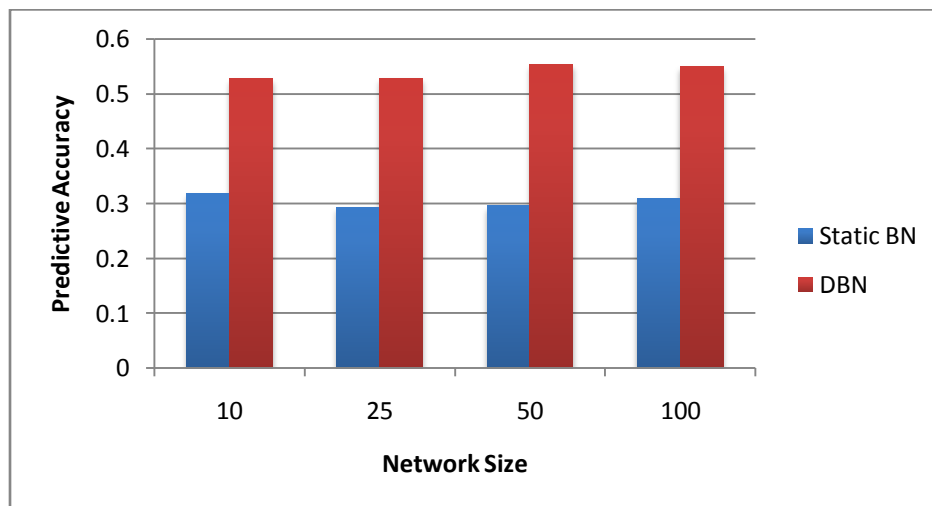


Figure 15: Accuracies of Static and Dynamic Structures

We can see clearly from this graph that the DBN model offers significant improvement over the static network on which it is based. What is also important to note is the relatively consistent accuracies of the static networks despite the increase in size. In fact, for the two medium sized networks the accuracy drops and the smallest network is the most accurate. This is because there are no temporal arcs and therefore no way to access the previous day's evidence.

We can still see an improvement over simple guess work as these networks contain constructed probability distributions which better reflect the nature of the environment than a simple guess. They also contain some dependencies which will improve the accuracy a little without the aid of evidence.

However, it is the addition of evidence that allows us to take advantage of the power of Bayesian Networks and we can clearly see this from these results.

6.6 Computational Costs

Within the scope of this project the main costs of both learning and inference is CPU processing rather than memory restrictions. So it is the computation time that we measure rather than the memory used.

The computational costs of learning are bound to be very high. Learning of Bayesian Networks has been shown to be computationally complex and finding the optimal structure is intractable. Even approximate search algorithms such as the one we have developed are computationally complex. Here we will analyse the time spent on computation during the learning algorithm, measuring the time spent on the various aspects of the algorithm at each iteration to find areas for potential improvement. We also compare the times for learning different sized networks to get an idea of the further scalability of the algorithm.

The learning algorithm contains two main aspects which require significant computation. The first is candidate generation, which involves selecting where to add the arc and then learning the new CPT table for the relevant node. The second is the scoring of the new candidate network, which mostly consists of inference. Several other functions are also performed during each iteration but these require relatively little computation.

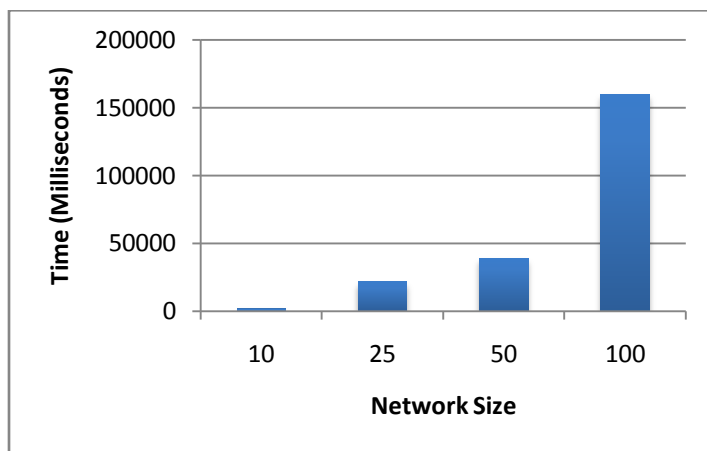


Figure 16: Average Time in Milliseconds for Learning Iteration

We can see from this graph the exponential increase trend in the total learning times. As the size of the network increases the learning time increases at an even faster rate. This is important to note when considering the construction of a far larger real world model. The time that would be needed to build such a model would be huge, even if very powerful supercomputers were used. However, the construction would be a once off cost and so the model it develops might still prove beneficial in the long run.

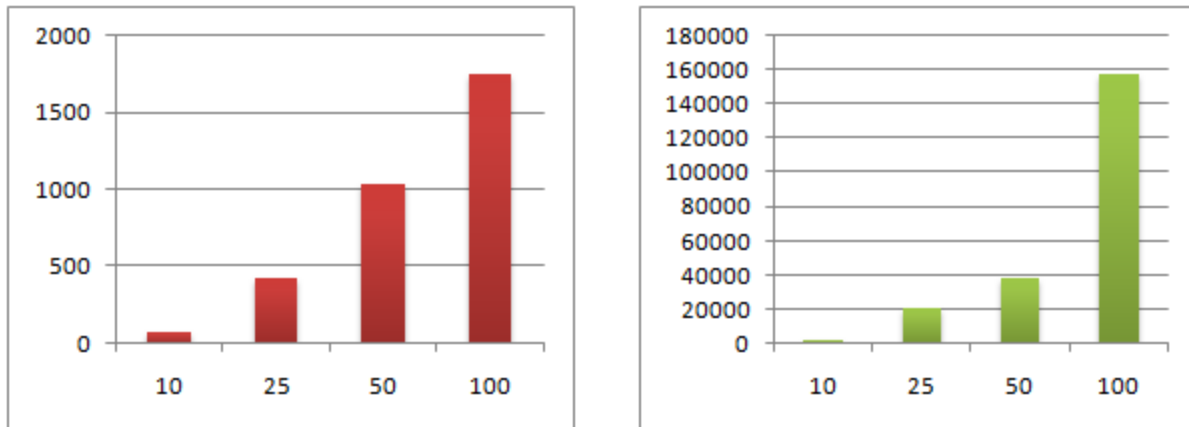


Figure 17: Average Time Spent on Generating a Candidate (Left) Average Time Spent on Scoring (Right)

The two graphs above show the learning times for the two main aspects of the learning algorithm. Together these two processes make up almost all of the time taken in each step of learning. As with the total learning time we can see the exponential trend although it is much more pronounced in the scoring function.

The scoring function is where the inference is performed and it is in this area that the effect of increasing the size of the network is truly felt. In fact we can see from the times spent on these two processes that the time spent on candidate generation is insignificant when compared to the time taken for scoring.

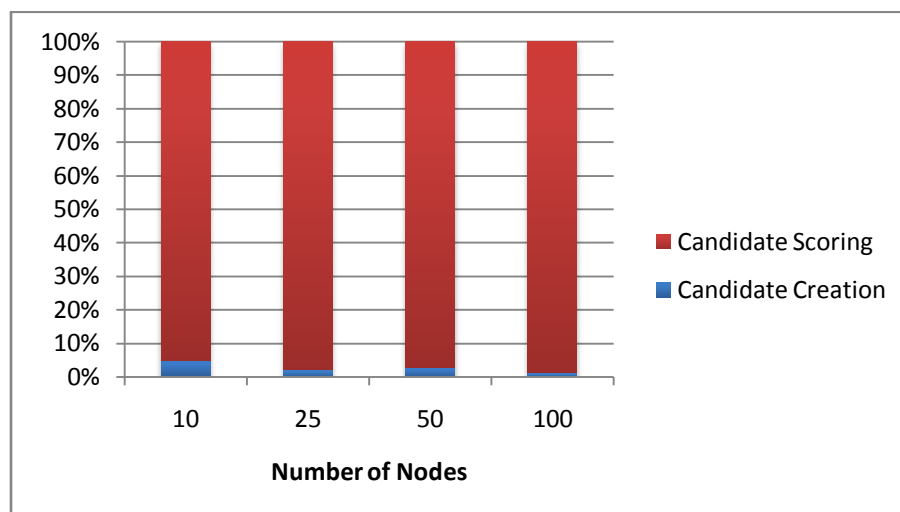


Figure 18: Ratio of Time Spent In Algorithm Sections

Since a candidate is chosen randomly and requires the learning of only a single CPT table this observation was expected. In fact the time spent on candidate generation for any individual is entirely independent of the size of the network; it is only dependant on the number of parents of the node. The average time does show an increase in this time with network size increase though. This is because in a

larger network there are more nodes to learn and so when the number of parents increases in the later stages the time taken here dominates the early stages more than in a smaller network.

Since the complexity of the network grows with every step in which we add an arc it is also important to notice how the time spent on each iteration increases over time. Below is a graph showing the time spent on each iteration for a hundred node network.

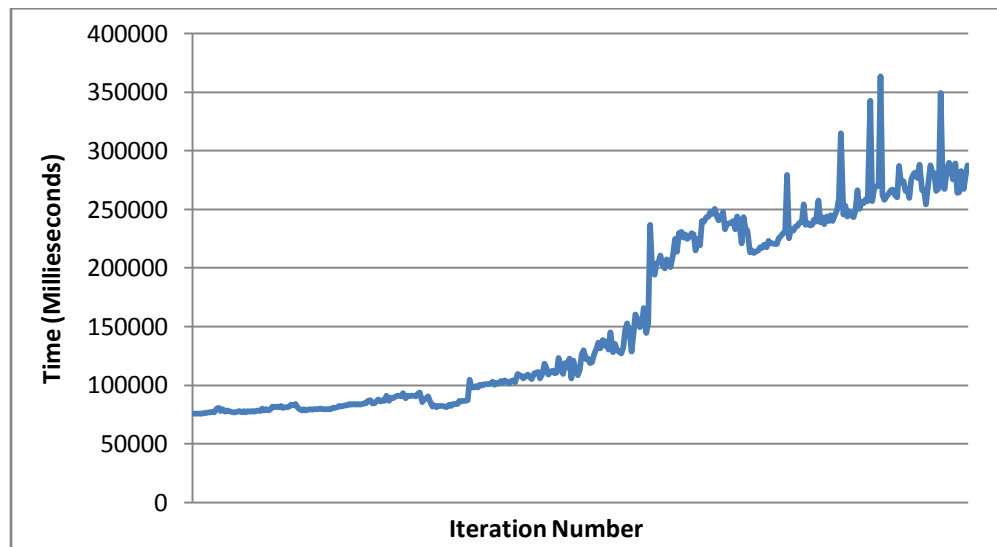


Figure 19: Iteration Times for Learning a Hundred Node Network

As we can see there is a general upward trend in the time taken for each iteration. As time passes the times also start to become more erratic. It is possible that this is due to the addition of an arc to an already complex node. We have noted that the complexity of a nodes CPT table increases by a factor of 5 with every arc added and so after several arcs are added this could result in some spikes. Also since in the later stages we typically accept fewer candidates the time of one step bears far less resemblance to the previous, unlike the early stages where every step is built on the last since most candidates will be accepted.

6.7 Increasing Degree of Temporal Forecast

Also of interest is our ability to generate accurate forecasts over a few days rather than just for the next day. Until now we have measured the accuracy and ability of forecasts and networks based on forecasts solely for the next day. We have inserted evidence into a single time-slice for each node and then made a forecast for each of those nodes in the next time slice.

What we aim to identify now is how effective our forecasts remain as they are extended further into the future and separated from the evidence we can provide. The major problem that hinders these further forecasts is that as we move further into the future we become separated from our evidence. When we are forecasting for just one data ahead we have all the evidence for the previous day. Now if we try to make a forecast for the day after the next, we have evidence for today, then we can make forecasts for tomorrow and simply extend our DBN by one more time slice to forecast an additional day.

However, as we extend the DBN further into the future the complexity grows very quickly. Even for the small networks extending the model past a few days made it too complex for the exact inference algorithms available in *Smile*. So for this experiment we were forced to use an approximate algorithm. Several of these are present in *Smile* along with the exact algorithms and Logic Sampling was the method chosen. The graph below shows the forecast accuracies for up to five days.

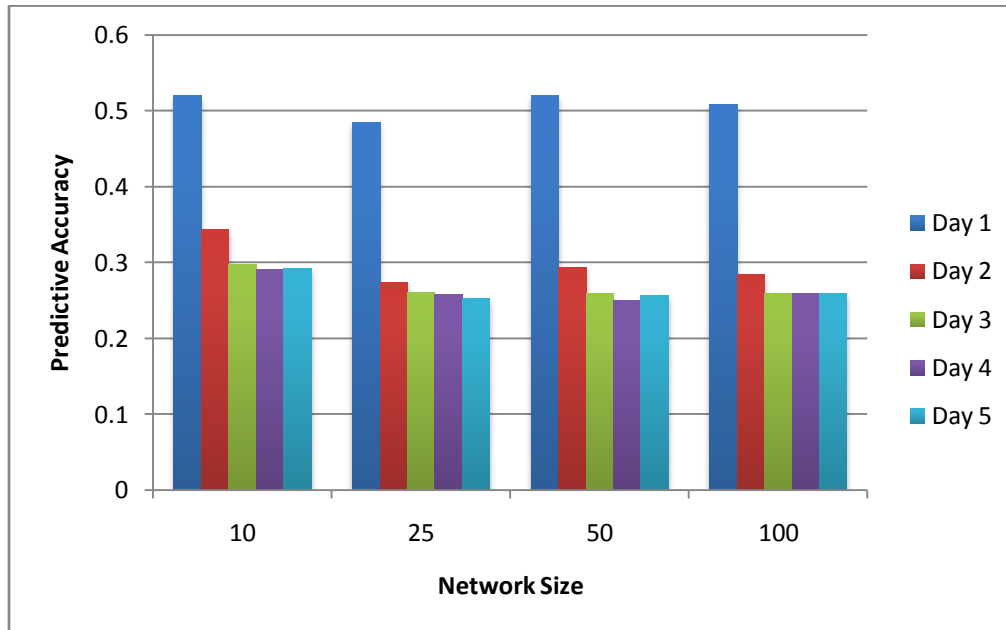


Figure 20: Forecast Accuracy for Different Temporal Degrees

Firstly, we note from this graph that our first days accuracy is different from our earlier results. The accuracy is substantially lower than before when exact algorithms were used. The results are also a little more erratic and in general seem to decrease as network size increases. These effects are entirely due to the use of the approximate inference algorithms and the cumulative effects as network size increases can be seen.

Apart from the use of an approximate algorithm we can also note the behaviour for greater temporal forecasts. The accuracy for the second day is much lower but for the ten node network is still better than the static network. However, for the larger networks the accuracy by the second day has already dropped to below the static network accuracies as can be seen in Figure 15.

In theory the behaviour as the temporal degree increases should eventually return to an equilibrium very similar to that of the static network. As we move further and further away from the evidence we are basically reducing its effect on the probabilities and so in essence returning to the state we had with the static network which was ignorant of any evidence.

We can see that from day three onwards we do reach the expected equilibrium; it is simply lower than expected. This reduction in accuracy is therefore most likely due to the change in algorithm as we can already see in part from the difference between our one day forecast accuracies with their originals. However, it is still an improvement over a pure guess.

7. Findings

This project set out with three major goals; to create a learning algorithm capable of building DBN's based on an initial static structure, to build DBN's capable of weather forecast and to optimize these networks for performance. During development less emphasis was placed on the final goal due to the complexity of the first two goals. Building networks took the focus, and the investigation into their potential accuracy would determine if these optimizations would be needed at all.

Some thought was given to the problem of optimization during the learning process. The function which determines if candidates are selected does consider and penalize more complex solutions and allows and even attempts to favour arc removal as a measure to reduce complexity. The learning algorithm selected has potential to include several measures to effectively reduce the complexity of the networks it creates although if large real world networks were to ever be built the problem of optimizations may once again come to the fore.

7.1 Learning Algorithm

Simulated annealing (SA) provided an effective solution, creating networks that were much improved over the static structure on which they are based. The ability of the algorithm to avoid local minima provides a key benefit while also running in a reasonable amount of time given the limited computational resources available.

Typically the algorithm showed the desired behaviour. Initial candidates were all given similar favour and as time passed more emphasis was placed on better solutions. However, as the T parameter approached zero asymptotically in the final stages the scores also tended to become too low in the very late stages. Some simple adjustments the annealing schedule could remedy this minor problem. The scoring function developed is also a little too wide in range. During its development the entire range of initial scores was considered. However, it is impossible that adding a single arc could improve the accuracy by 100%. In fact the difference between candidates is never more than a few percent and typically much less than this. The function could focus more on this smaller area.

We can also see that as time passes the benefit gained from adding more arcs is also reduced. The reasons for this are twofold. Obviously in the very early stages every arc, even if not embodying a strong causal link means that the Network can see more of the past days evidence and so will provide a strong benefit. As time passes this effect is reduced. In addition as time passes and the number of parents for each node grows the sizes of the CPT's grows. As the CPT's grow our ability to accurately capture any causal dependencies is reduced due to our limited data volume. This means we see less of the potential benefit of each arc even if it could have a much larger effect.

In many cases the networks generated within this project tended towards becoming too complex, possibly adding arcs which provided some gain but not enough to possibly justify its inclusion. This is a problem that simulated annealing can help to prevent. Since less optimal solutions can be accepted arc removal which slightly lessens the accuracy becomes a viable solution. The SA algorithm which was used simply needs to place more emphasis on arc removal, especially as more time passes.

This is where the main benefit of using an SA approach would be seen. In typical problems SA allows less optimal solutions to be selected to avoid local minima. Within a BN application it is the arc removal that typically results in less optimal solutions but it alternatively means the networks is less complex. A less complex network is possibly more globally optimal since it requires less computation. Our SA approach allows for these solutions to be selected meaning less complex networks can be created. The problem is simply that not quite enough focus is placed on arc removal in the current algorithm.

We can also see from the analysis of the computational complexity that the time spent on generating a new candidate is almost insignificant when compared to the time spent on scoring. In fact it makes up less than 3% of the total learning time in most cases. Given this it may be to our benefit to spend more time on this stage and attempt to select candidates which have more potential to benefit our network. For instance when we select nodes to connect with an arc we could analyse their current complexity or even geographical positions to gain some insight into the potential of the arc. By generating a few candidates and selecting the one which shows some potential over the others could result in much improved efficiency and possibly even reduced complexity.

Another aspect on which little focus was placed was the penalty for adding complexity or the bonus for removing it. The solution implemented was to simply increase or decrease the score by a factor. Relating this penalty to the current T value could improve its effectiveness. For instance, in the early stages the penalties should be minimal as we simply want to get the accuracy up and establish a base solution. As more time passes the penalty or bonus could increase essentially making the algorithm more greedy in the aspect of an efficient solution.

This together with the approach getting more greedy with regards to accuracy as well in the late stages could result in a solution which rapidly approaches a good solution late on.

In fact the only major area in which the SA algorithm developed that requires improvement is the candidate generation. A better solution which lessens the search graph width and places more focus on removing redundant arcs in the later stages could vastly improve the overall effectiveness and efficiency of the algorithm and its solutions and provide a good intermediate solution for optimizing the created networks.

However, the key problem with learning was related to the volume of data used. The training set used to build all the networks contained 602 days of data. When it came to learning the CPT's this was simply too small a set. The nodes in the static networks already contained one or two arcs at least and by the time a second or third temporal arc was added the sizes of the CPT tables was at least as big as the training set itself and in some cases 5 or even 25x times as large for some nodes. In these cases the parameters could not be calculated with any real accuracy and simply defaulted to the uniform prior which we began with.

Using a larger volume of data for the learning of the CPT's will no doubt vastly improve both the effectiveness of learning and also the effectiveness of any created networks. Dependencies can be more easily captured if more data is available. Alternatively a new approach must be used for parameter learning which requires less data.

7.2 Forecast Ability

We can clearly see the benefit of DBN's from the results. The ability to add the evidence of yesterdays observed values almost doubles the accuracy of our one day forecast. Given some of the problems we have already mentioned with the CPT's this result seems to point towards DBN's being a potentially good solution.

What we must also consider though is the number of states we are forecasting for. As we discussed earlier we have allowed our variables to take on five potential states. This means that in the worst case with no domain knowledge we should be predicting 20% of the values correctly. The static networks produced in the other learning section of the project predicted, with no evidence, roughly 30% of the values correctly. By adding in temporal arcs and allowing evidence to be input we have increased this to around 55% on average. This is an improvement of nearly three times over a plain guess.

Temporal Forecast

The results for increased temporal forecast degrees are disappointing. Only the smallest network showed an improvement over the static network for the second day and in all other cases the forecasts were lower. This is most likely due to the use of approximate inference algorithms but none the less we can see that forecasts for the second day will at best offer a slight improvement over the static network.

Past the second day we can see from these results that the accuracy drops into a state of equilibrium as we expected. For any of these days the static network will provide at least as good a forecast and so the complexity of the increased DBN size is not warranted.

With better learnt CPT's these accuracies may improve especially for the first and second day. We can see from the later forecasts though that the removal from the evidence clearly means that forecasts past a few days will not show much improvement. To improve these accuracies other options, such as ensembles must be explored.

7.3 Related Work

The *Hailfinder* system was the first BN based system designed to forecast weather. In comparison to the system we have developed it is highly geographically constrained. The *Hailfinder* model was designed to forecast severe weather in the North Eastern Colorado region and is thus highly specific in its interpretations of any variables and forecasts. In comparison our model is general, searching for dependencies over any arbitrary region.

The techniques used for developing the *Hailfinder* model are also in stark contrast to those we used. The BN used in the *Hailfinder* model, both the structure and the parameters, was developed through expert elicitation. This meant that the construction process was long and involved, taking over two years, despite its constrained nature. Our networks were developed entirely through fairly general machine learning algorithms. This means larger networks can be learnt with far greater ease.

Since the *Hailfinder* system was entirely built by expert judgement there was also a greater need for the simplification of the network to allow the parameter learning task to become manageable. Less focus

was placed on this task in our project although the learning algorithm does contain techniques for reducing complexity.

The primary difference between *Hailfinder* and our system is that their system models the dependencies between several different variable types and generates specific output from specific forecast nodes. The *Hailfinder* model allows for the input of several variables such as vertical motion, cloud cover, atmospheric moisture and others as well as some expert judgements on the atmospheric state and uses this evidence to produce specified outputs for the five regions it covers.

In the model we have developed there is no distinction between input or output nodes. Instead each node represents a specific geographic point and we input data for each one in the first time slice and forecast in the second. Each network contains a specific variable either max or min temperature or precipitation. Our model is based more on temporal forecasts of one variable rather than the *Hailfinder* model which attempts to forecast a particular occurrence of severe weather given the current atmosphere state.

Unfortunately no evaluation of *Hailfinder* was ever carried out. Collecting observational data is a costly process and since the model was not developed in time to compete in the forecast shootout competition for which it was developed, no meaningful testing or adjustments were ever made to the model. This means we cannot give a comparison of the resulting accuracy of their system. However, the techniques and areas of the application of their model are sufficiently different to make a meaningful comparison difficult at best.

The system developed by Cano et al [2004] bares more similarity to our system. Like ours it uses a BN to represent the spatial dependencies between various weather stations within an area of interest. The original model [Cofino 2000] built a BN which included 100 stations as well as nodes for numeric model forecasts which served as the evidence nodes for their system. Using a nowcasting approach for validation in which evidence is input for some nodes and the accuracy of the remainder tested, they successfully explored how BN could be applied to meteorological problems.

Our system follows the same approach of modelling spatial dependencies but extends their ideas by creating a DBN which can be used for temporal forecast. Unlike their system which uses input from a numeric model as input to a static network to forecast station values, our system uses observed values from stations to forecast future values for those same stations.

The number of stations used was the same in both our investigations and the number of states to which the data was discretized was also similar (5 in our as opposed to 4 in theirs). The primary difference was in the amount of data used for each station. In their learning process over 40 years of data was used compared to less than two years in our learning algorithms. This brings into context the problems which we have discussed in learning the CPT tables. Given the comparatively minute size of our training set we simply could not learn these CPT parameters effectively.

Unlike *Hailfinder* the systems developed by Cano et al [2004] is dependent on machine learning algorithms to build both its graphical structure and learn the CPT parameters. They used the K2 learning

algorithm which was used in part in the causal modelling part of the paper although the networks discussed in this paper are based on static structures learnt using the greedy thick thinning approach.

Currently only tentative research has been carried out into the use of BN's for weather forecasting. Cano et al [2004] have explored how a variety of problems, especially those associated with spatial consistencies might be solved using BN's. They did explore their use for weather forecast although this was more along the lines of downscaling the forecasts of a numeric model to generate more local forecasts. The *Hailfinder* system [ABRAMSON 1996] was developed to explicitly generate forecasts of severe weather. However, it covered only a small area and was never tested.

Our project has extended this research by exploring the use of DBN's and machine learning algorithms for the explicit use of weather forecast. While our research has the similar problem of being both relatively constrained in size the early results show some promise for the use of DBN's.

7.4 Lessons Learnt

Through the course of this project the author has learnt many valuable lessons. Research, group work and problem solving on a project of this size and complexity has highlighted the areas in which the author has ability as well as those areas in which the author needs to work harder.

Working for an extended period of time as part of a team has many challenges. Keeping communication open and effective is always a challenge but by keeping regular meeting and providing input on each other's work we have maintained an effective system. Being able to listen to others ideas about you own work and being able to input ideas into their own work keeps your mind fresh and allows you to build new ideas and not get lost down pointless paths of work.

Researching and building a large system is also filled with challenges. Initially the author had little working knowledge of Bayesian networks. Extensive research of the subject has provided the author with the experience and knowledge required to learn new ideas and complex theories. Initially research was approached from the wrong angle, searching for research papers on topics similar to our own. Because there was little foundational knowledge of BN's much of this work was either not understood or worse misunderstood.

After stepping back a little and obtaining a better working knowledge of the theory involved this work became much more accessible although in some cases it still remained beyond the current knowledge of the author. Understanding the approaches needed to conduct research is an invaluable tool.

Time management is also a crucial skill. Having to work on mostly your own time for a long period can easily lead to lethargy. Although the project always remained on track and was sometimes even ahead of schedule there were still times when a little of this lethargy was felt by the author. However, tackling this problem has provided excellent experience into how to manage time. At times during the coding when the author simply felt unable to stare at and debug code anymore, focus was instead shifted to writing this report for a little while. And when report writing became the focus later on and too became a chore at times a little work on making some images or tweaking code provided a release.

In all, the project was always on track and the report itself seemed well ahead of schedule. The approach of writing the report as work on the project was done proved its worth by highlighting which problems needed to be tackled on all fronts and showing where extra work was needed.

Of course the greatest ability gained from this project has been the confidence gained from working on a complex project. Building your knowledge of a complex subject and being able to understand and construct solutions of your own conception gives you the confidence to tackle larger problems and the author has definitely benefited from this confidence.

Hindsight of course provides us all with perfect vision. If the project could be repeated the author would have focused more energy into the basics in the early stage. Research into the deeper complexities should have been left until this basic understanding was gained and when those areas were better acknowledged as being within the scope of the project.

A more aggressive approach to the early programming might have also allowed for more time to let the algorithms run. This would have meant a greater portion of time could have been given to tweaking the algorithm. We could have also built several networks which would have allowed for us to gain better insight into how well the accuracy of created networks remains over several learning attempts.

However, I think the overall approach to the project was good. The methodologies used were the right ones and helped to keep work on track.

8. Conclusions

The networks developed within this project have showed some promise if not quite managing to fulfil early expectations. Despite the use of a limited data set and the problems this meant for learning parameters we have constructed networks which show much improved accuracy over their static counterparts. It is as we extend our forecasts further into the future that the results become disappointing. Forecasts for a second day show very little benefit although part of this may be due to the use of approximate inference algorithms. Past the second day we can see even at this stage that the use of DBN's does not show much ability to generate forecasts.

However, this is not an immediately discouraging result. Forecasts of three or more days into the future have proved an exceptionally difficult problem even for very complex well developed systems. There also exist potential areas which may be explored to improve long term accuracy but which fell beyond the scope of this project.

Given these early results DBN's do show promise for weather forecast. With a limited training set and relatively small network sizes we have successfully managed to forecast over 50% of the data items correctly. With a more fully explored solution this accuracy will improve greatly and by including other weather variables into the model a complete forecast engine could be developed that could show good accuracy while still maintaining computational tractability.

The use of simulated annealing also shows promise as a DBN learning algorithm. Given the effectiveness of the created networks it has proven itself to be good approach and with greater exploration into its ability to limit complexity it could be easily adapted to build both accurate and efficient solutions.

The single biggest problem remains the limited size of the training set and the effect it has on the learning of CPT tables. There is little doubt that with a larger training set or more effective parameter learning algorithm the networks created would be far more accurate and the efficiency and ability of the learning algorithm would be improved.

Until networks can be built with larger datasets we will not be able to gain full insight into the potential of DBN's for weather forecast. At this early stage this approach shows some promise but a full investigation is required before we can assess their true ability.

9. Future Work

9.1 Multiple Weather Variables

One key extension that could provide improvements in forecast capability is to include multiple variables in a single DBN. Our DBN's have modelled a single variable, such as temperature or precipitation in each network, mapping out the dependencies between stations.

The models might be extended to include many variables in a single model, mapping not only inter-station dependencies but also the dependencies of different variables on each other. However, this is not a straightforward extension. By including a range of different variables the already super-exponential search space for the optimal graph structure is increased at a rate which fast makes learning impossible without the inclusion of new restrictions.

It is very possible though to place restrictions to allow for efficient learning though. Several ideas may provide us with possible methods for controlling the size of the search space. For instance, the inter-variable dependencies could be limited to a single station at a time, and a technique similar to that of *Hailfinders* [ABRAMSON 1996] scenarios could be used to map inter-station dependencies based on the inter-variable predictions for that state. This would limit the complexity to a similar order of magnitude to that of the current Networks.

9.2 Simulated Annealing Improvements

As was discussed earlier, the SA learning algorithm is to a degree limited by the diameter of the search space graph. If a different candidate selection method could be developed which limits the diameter of this graph from $O(n^2)$ size, the effectiveness of the learning algorithm could be dramatically increased. The candidate selection also needs to place more focus on arc removal especially in the later stages when the number of potentially redundant arcs has grown. This will allow the algorithm to take advantage of the key benefit of SA's ability to avoid local minima by accepting slightly weaker candidates.

Some minor adaptations to the scoring function could also improve the algorithm dramatically. First the scoring function needs to focus more on the range in which candidates typically fall rather than the maximum possible range. This could be done with some alteration the T value. It basically represents the variance in the normal distribution within our function. It needs to start off with a much lower value for instance 0.25 rather than 1.0. This would focus the scoring function a lot more tightly on the useful range. If T also approaches 0 slightly slower in the late stages we would also see an improvement.

Finally, the algorithm should take steps to avoid complexity more often. The penalty system is independent of T in the current function. By linking it to the T value we could shift the focus more onto better improvements or arc removal later on. The candidate selection could also take into account the number of parents of the selected node when adding or removing arcs.

9.3 Optimization

One of the initial goals of the project was to also apply a series of potential optimization algorithms on any created networks. While the learning algorithm developed did penalize additional complexity in new candidates, little work was performed on optimizations. This was mainly due to some time constraints and the complexity required in implementing some of the techniques.

However, optimizing the networks created using some of the ideas from the Hailfinder system [ABRAMSON 1996] or other such as node clustering or scenarios the efficiency of the graph could be improved and possibly allow for the number of variables contained in the network to be increased, allowing for possibly more accurate predictions.

The networks created are also very complicated with many temporal arcs existing for each node. More emphasis could be placed on arc removal and less favour given to small improvements during the learning process.

9.4 Greater Temporal Dependence

The algorithm can easily be extended to allow for arcs of increased temporal degree. Allowing for the network to find inter-node dependencies of a higher temporal degree may improve the long term forecasts. However, this would increase the network complexity and in practice many of these dependencies might have already been embodied in arcs of lower temporal degree.

Increasing the maximum temporal degree would also allow us to increase the number of days we may use as evidence. If we only have degree one temporal arcs then adding a second day of evidence after the first means all the evidence from the first day is absorbed by the second and has no effect on our inference. However if second degree temporal arcs did exist, then this evidence may provide some benefit to our forecasts.

9.5 Parameter Learning

Another aspect of the learning process that could be greatly improved is the efficiency of the parameter learning. The algorithm used, while being simple was constructed in a way that meant the dataset had to be read through entirely for each entry in the CPT. This algorithm could be improved with the use of a more complex data structure to read through the data set only once.

In addition, other algorithms which take advantage of parameter dependencies could be used to provide further improvement.

Work on the parameter learning was highly constrained due to limited time. Given that the project focus was more biased towards the creation of effective networks rather than focusing on the improvement of the learning algorithm less time was spent on improving these aspects.

The primary problem with the parameter learning was the relatively small training set. This meant that as the number of parents grew as was inevitable given that both normal and temporal parents were added there was simply not enough data to learn the CPT's effectively. This significantly hampers the effectiveness of any forecasts and so testing the project with much larger datasets may show that DBN's are in fact far more accurate than this project has managed to show.

9.6 Increasing Accuracy of Long Term Forecasts

We have seen that past two days we do not see a benefit to forecast accuracy from using DBN's. However, there are several approaches that may be explored to improve these accuracies.

First, we have already discussed the possibility of increasing the temporal degrees of the arcs used in the network. Not only will this mean that we can add in more evidence to increase the accuracy it will also mean that the effects of some evidence will penetrate further into later time slices. Our second day's forecasts will no longer have a day of predictions in between them and the evidence if we have some second degree temporal arcs. Rather some nodes will be directly connected to nodes which contain evidence. This has great potential to increase the accuracy of forecasts past one day but again it will mean increased complexity.

We could also use an ensemble approach. Rather than simply increasing the number of time slices we could simply use two slices and the rolling window approach typically used in complex solutions. Once we have made our first days forecasts, these forecasts could be entered as evidence for the second day and so on. Whether this will offer significant improvement to accuracy will need to be further explored but it will also reduce the complexity allowing for the possible use of exact inference algorithms again.

Bibliography

- ABRAMSON, B., BROWN J., EDWARDS W., MURPHY A., AND WINKLER R.L. 1996. Hailfinder: A Bayesian System for Forecasting Severe Weather. *International Journal of Forecasting* 12, 57-71.
- BERLINER M.L., ROYLE J.A., WILKE C.K., AND MILLIFF R.F. 1998. Bayesian Methods in the Atmospheric Sciences. *BAYESIAN STATISTICS* 6, 83-100.
- CANO R., SORDO C., AND GUTIERREZ J.M. 2004. Applications of Bayesian Networks in Meteorology. *Advances in Bayesian Networks*, 309-327.
- COFINO A.S., CANO R., SORDO C., AND GUTIERREZ J.M. 2002. Bayesian Networks for Probabilistic Weather Prediction. In *Proceedings of the 15th European Conference on Artificial Intelligence*, IOS Press, 695-700.
- COELHO C.A.S., STEPHENSON D.B., DOBLAS-REYES F.J., BALMASEDA M., GUERRER A. AND VAN OLDENBORGH G.J. 2006. A Bayesian approach for multi-model downscaling: Seasonal forecasting of regional rainfall and river flows in South America. *Meteorology. Appl.* 13, 73–82
- DE CAMPOS L.M., AND CASTELLANO J.G. 2007. Bayesian network learning algorithms using structural restrictions. *International Journal of Approximate Reasoning* 45, 233–254
- HECKERMAN D., GEIGER D., AND CHICKERING D.M. 1995. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning*, 20, 197-243.
- LYNCH P. 2008. The origins of computer weather prediction and climate modelling. *Journal of Computational Physics*, Volume 227, Issue 7, 3431-3444
- MCMILLAN N., BORTNICK S.M., IRWIN M.E., AND BERLINER L.M. 2005. A Hierarchical Bayesian Model to Estimate and Forecast Ozone through Space and Time. *Atmospheric Environment* 39, 1373-1382.
- MURPHY A.H., AND WINKLER R.L. 1984. Probability Forecasting in Meteorology. *Journal of the American Statistical Association*, Vol. 79, No. 387, 489-500
- PEARL J. 1996. Decision Making Under Uncertainty. *ACM Computing Surveys*, Vol. 28, Num. 1.
- FRIEDMAN N., MURPHY K., AND RUSSELL S, 1998. Learning the Structure of Dynamic Probabilistic Networks. *Proc. Fourteenth Conference on Uncertainty in Artificial Intelligence*.
- MURPHY K.P. 2002. Dynamic Bayesian Networks: Representation, Inference and Learning. Thesis Paper. University of California, Berkeley.
- KORB K., NICHOLSON A.E., 2003. Bayesian Artificial Intelligence. CRC Press.
- HULST I. R. J. 2006. Modeling Physiological Processes with Dynamic Bayesian Networks. Thesis Paper. University of Pittsburgh.

Appendix 1: Glossary of Terms

DBN – Dynamic Bayesian Network
SA – Simulated Annealing
BN – Bayesian Network
PA – Predictive Accuracy
IR – Information Reward
IR_B – Bayesian Information Reward
CPT – Conditional Probability Table
DAG – Directed Acyclic Graph

Appendix 2: Sample Test Networks

2.1 Simple Hand Calculated Example Networks

These small networks, developed by hand were used to test some simple elements of the system for logic errors. Given their small size and a small associated data set, parameters and other variables could be calculated by hand using the counting algorithm and used to compare with generated results.

2.1.1 Three Node Generic Network

This simple network was used in initial parameter learning tests to test the static parameter learning algorithm. It was also used in early tests of the structure learning algorithm to test if nodes were correctly updated to temporal plate nodes.

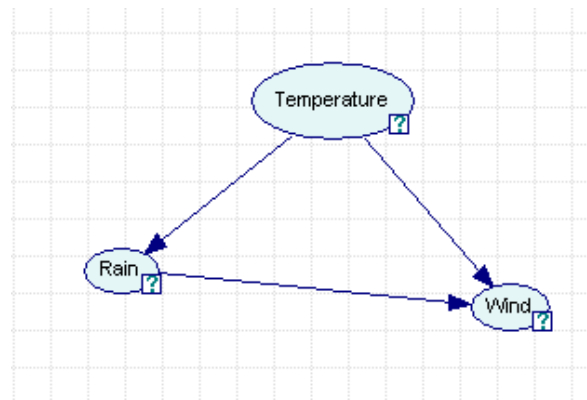


Figure 21: Three Node Generic Network

Table 1: Dataset for Three Node generic example

Temperature	2	1	2	2	1	0	1	1	0	1	2	2	1	2	2
Rain	0	1	0	0	1	1	1	0	0	1	0	0	0	1	0
Wind	2	2	0	1	3	1	2	3	3	1	0	0	1	2	1

Table 2: CPT for Temperature Node

Cold	0.13333
Cool	0.4
Hot	0.46667

Table 3: CPT for Rain Node

Temperature	Cold	Cool	Hot
No	0.5	0.33333	0.85714
Yes	0.5	0.66667	0.14285

Table 4: CPT for Wind Node

Temperature	Cold		Cool		Hot	
	No	Yes	No	Yes	No	Yes
None	0	0	0	0	0.5	0
Low	0	1	0.5	0.25	0.33333	0
Medium	0	0	0	0.5	0.16667	1
High	1	0	0.5	0.25	0	0

2.1.2 Four Node Temporal Example

This example was constructed to test the temporal parameter learning algorithm. It is similar in concept to the above network but also contains temporal arcs.

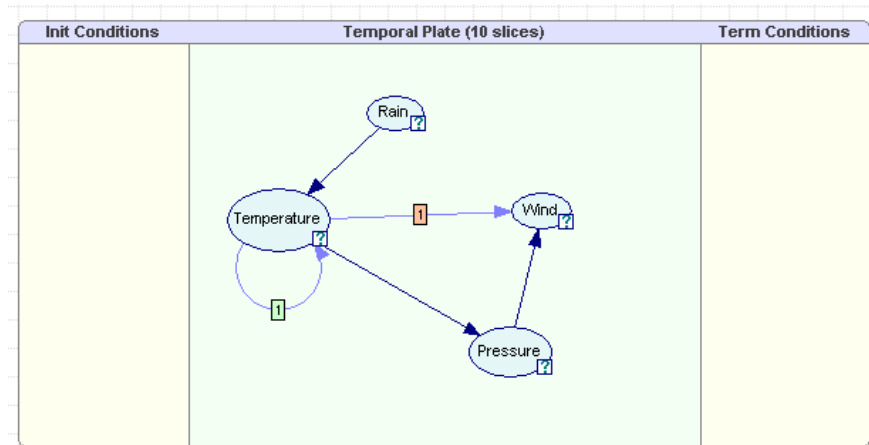


Figure 22: Four Node Temporal Network

Table 5: Dataset for Four Node Temporal Example

Rain	1	0	0	0	1	1	1	1	0	0	1	1	0	1	1
Temperature	1	2	1	0	2	1	0	0	1	2	1	0	1	2	1
Pressure	0	0	1	1	0	1	0	0	1	0	1	1	1	0	1
Wind	0	1	1	2	0	3	0	2	0	1	1	1	3	3	2

Table 6: CPT for Rain Node

No	0.6
Yes	0.4

Table 7: CPT for Temperature Node for T=0

Rain	No	Yes
Cold	0.166667	0.333333
Cool	0.5	0.444444
Hot	0.333333	0.222222

Table 8: CPT for Temperature Node for T=1

Rain	No			Yes		
Self - 1	Cold	Cool	Hot	Cold	Cool	Hot
Cold	0	0.333333	0	0.5	0.666667	0
Cool	1	0	1	0	0	1
Hot	0	0.666667	0	0.5	0.333333	0

Table 9: CPT for Pressure Node

Temperature	Cold	Cool	Hot
High	0.5	0.8572	0
Low	0.5	0.1428	1

Table 10: CPT for Wind Node for T=0

Pressure	High	Low
Very Low	0.125	0.4285
Low	0.375	0.2857
High	0.25	0.1428
Very High	0.25	0.1428

Table 11: CPT for Wind Node for T=1

Pressure	Low			High		
Temperature-1	Cold	Cool	Hot	Cold	Cool	Hot
Very Low	0.5	0.25	0.25	0.5	0	0
Low	0	0.5	0.25	0	0.5	0.5
High	0.5	0	0.25	0	0.5	0.25
Very High	0	0.25	0.25	0.5	0	0.25