# Brock University

Department of Computer Science

**Evolving Dynamic Bayesian Networks with Multi-objective Genetic Algorithms**

B.J. Ross and E. Zuviria
Technical Report # CS-05-03
May 2005

Brock University
Department of Computer Science
St. Catharines, Ontario
Canada L2S 3A1
www.cosc.brocku.ca

# Evolving Dynamic Bayesian Networks with Multi-objective Genetic Algorithms

Brian J. Ross [1]

*Brock University, Dept. of Computer Science*
*St. Catharines, Ontario, Canada L2S 3A1*
*bross@brocku.ca*

Eduardo Zuviria [2]

*Brock University, Dept. of Computer Science*
*St. Catharines, Ontario, Canada L2S 3A1*
*zuviria@cs.queensu.ca*

**Abstract**

A dynamic Bayesian network (DBN) is a probabilistic network that models interdependent entities that change over time. Given example sequences of multivariate data, we use a genetic algorithm to synthesize a network structure that models the causal relationships that explain the sequence. We use a multi-objective evaluation strategy with a genetic algorithm. The multi-objective criteria are a network's probabilistic score and structural complexity score. Our use of Pareto ranking is ideal for this application, because it naturally balances the effect of the likelihood and structural simplicity terms used in the BIC network evaluation heuristic. We use a simple structural scoring formula, which tries to keep the number of links in the network approximately equivalent to the number of variables. We also use a simple representation that favours sparsely connected networks similar in structure to those modeling biological phenomenon. Our experiments show promising results when evolving networks ranging from 10 to 30 variables, using a maximal connectivity of between 3 and 4 parents per node. The results from the multi-objective GA were superior to those obtained with a single objective GA.

*Key words:* dynamic Bayesian networks, multi-objective optimization, genetic algorithms

[1] Corresponding author.
[2] Current address: Dept of Computing, Queen's University, Kingston, ON, Canada

## 1 Introduction

A Bayesian network (probabilistic network, belief network, probabilistic graphical model) is a compact probabilistic representations of entities exhibiting complex interdependencies [5] [32] [36]. A Bayesian network (BN) takes the form of a directed acyclic graph, in which nodes represent observable and possibly hidden variables, and links denote causal relationships between variables. Once a BN is constructed to model some phenomenon of interest, it can be used for backward inference and approximate reasoning. For example, given some observed values of variables, the BN can be used to infer likely causes of these values, according to the relationships described in the network. Since exact inference with BNs is NP-hard, approximation algorithms are often necessary to obtain results. BNs have been applied to a variety of applications, for example, expert systems [4], bioinformatics [27], speech recognition [31], data compression [11], and evolutionary search optimization [33].

Whereas a BN represents the static relationships of a set of entities, a dynamic Bayesian network (DBN) models the relationship of entities that change state over time [14] [29] [36]. For example, genetic networks in biology are conveniently modeled by DBNs, since the influences of various genes upon one another are determined by sequences of phases that progress over time [22] [24] [34] [46] [47]. A DBN takes the form of 2 directed acyclic graphs. A prior graph denotes the static relationship of variables before the commencement of time. A transition network denotes the dynamic relationships that occur synchronously over time. To perform inference over time, the transition network is unwrapped or unfolded at each time step. In this way, a DBN can be used to infer over arbitrarily long time sequences.

Besides the use of BNs and DBNs for inference, another application is the study of the actual structural topology of a network. By examining the links connecting nodes, it is possible to ascertain explanatory rules regarding the relationships between variables. This is a form of data mining, in which a network can be used as a means of obtaining rule-based knowledge about a domain. The study of network structures is perhaps more important with DBNs, because probabilistic inference itself becomes less meaningful when considering phenomena that involve hundreds or thousands of changes over long time sequences. The probabilistic nature of the network is still of use, however, as it can be exploited by machine learning algorithms for obtaining accurate network structures from examples of the phenomenon of interest.

Machine learning algorithms for both BNs and DBNs have been extensively studied [3] [19] [7] [39] [17] [27]. The determination of an optimal BN to model given behaviour is an NP-complete problem [9] [6]. Learning algorithms are confounded by the fact that there are two interacting characteristics of net-

works that can work against each other – the inferred probability values from a network, which indicates how well the network fits the data, and the structural complexity of the network. A highly connected network is more likely to capture causal relationships, and hence will produce inferences with higher probabilities. For example, a maximally connected network denotes all possible causal relationships between the variables. Such a network, however, is not a likely model of the phenomenon of interest. Therefore, one would like a network to be accurate and structurally simple.

Consequently, the degree of connectivity of networks must be controlled in BN/DBN learning environments. Learning algorithms are naturally biased by the means that they evaluate candidate networks in terms of accuracy and simplicity. Many algorithms use a variation of the minimal description length (MDL) heuristic, which evaluates a network in terms of both probabilistic likelihood and structural simplicity [2]. However, because the MDL essentially uses a weighted sum to combine these factors, it can require user "tweaking" of weights to indicate the degree to which structural simplicity should balance with probabilistic likelihood. Since these factors are dependent upon the nature (size, complexity) of the training data, it is extremely difficult to predict the form of resulting networks for some new problem and data. An untweaked MDL score will likely produce overly connected networks for many problems, and especially for many real-world applications that are most naturally modeled by sparse networks. For example, biological phenomenon typically require low-connectivity networks with a maximum parent count of 4. Hence it is a reasonable goal to design a learning methodology that permits intuitive user definition of the structure of the solution networks.

This papers main contribution is to evolve DBNs using a multi-objective genetic algorithm. We use the basic BIC scoring scheme to evaluate candidate networks. Pareto rankings are ideal for separating the BIC heuristic's likelihood and structure terms during fitness evaluation. This circumvents the significant problem of finding a suitable weighting scheme for combining these factors, as is necessary with MDL evaluation. We introduce a very simple structural scoring heuristic in place of the more complex structural evaluation used in BIC evaluation. We use a straight-forward and intuitive means for controlling network structural connectivity, by adapting a chromosomal representation used in other GA work with genetic coding that is biased towards sparse, low-connectivity networks, which are common when modeling biological phenomenon. To test our approach, we performed experiments using multi-objective Pareto ranking with chromosomes having between 3 and 4 inward links per node, for target networks having between 10 and 30 variables. The results are promising, and are superior to those from single objective GAs using a weighted sum for likelihood and structural scores.

The paper is organized as follows. Some background on DBNs, evolutionary

computation, and multi-objective search is given in Section 2. The experimental setup is described in Section 3. The results are presented in Section 4. A discussion of the experiments is given in Section 5. Directions for future work conclude the paper in Section 6.

## 2 Background

### *2.1 Dynamic Bayesian Networks*

We are interested in modeling the behavior of a set of discrete random variables $X_1, ..., X_N$ during a sequence of time slices $t = 0, 1, 2, ...T$. The value of variable $X_i$ at time $t$ is denoted $X_i[t]$.

The BN describes a probability distribution over variables of interest. A Bayesian network (BN) consists of a directed acyclic graph G whose nodes correspond to the variables $X_1, ..., X_N$, and a set of parameters $P$ that encode the conditional probablity table associated with each node. These tables encode, for all variables, the probability that variable $X_i$ takes value $k_i$ given that a parent node takes the value $j_i$:

$$Pr(X_i = k_i | Parent(X_i) = j_i).$$

A dynamic bayesian network (DBN) extends the notion of a BN by incorporating a time element. A complete DBN consists of two components: (i) a prior network $DBN_0$ that specifies a probability distribution over the initial states (t=0); (ii) a transition network $DBN_T$ that specifies the transition probabilities $Pr(X[t+1]|X[t])$ for all variables $X$. When a prior network is supplied, it is used to denote the initial relationships between variables before the clock proceeds. The transition network is used during the time sequence, by "unwrapping" the network for each time slice. Hence it can denote a possibly infinite time sequence of events. The joint distribution over all the variables for time $t = 0, ..., T$ is:

$$DBN_0(X[0]) \cdot \prod_{t=0}^{T-1} DBN_T(X[t+1]|X[t])$$

for all variables $X$. As will be discussed later, our training data is generated so that the prior network has no influence on the probabilistic analysis of the transition network.

It is possible to discard the use of an explicit prior network, and use the transi-

tion network by itself. Doing so results in a more compact representation, but at the loss of probabilistic accuracy in denoting initial probabilistic influences of variables, which may not correspond to what is denoted in the transition network. In the case of this paper, the transition network alone is of interest, and the prior network is not used. Hence the joint distribution becomes:

$$\prod_{t=0}^{T-1} DBN_T(X[t+1]|X[T])$$

for all variables $X$.

### *2.2 Evolutionary Computation and DBNs*

Evolutionary computation has been widely used to synthesize BNs [25] [38] [12] [30] [10] [21] [18] [45] [43]. Two representative examples are the following. Larrañaga *et al.* use GAs to evolve BN structures [26]. Their representation is an adjacency matrix, and they transcribe the matrix using variable ordering rules to assure acyclicity. Their evaluation heuristic is a variation of the K2 algorithm [9], specialized to limit the number of parent nodes in considered graphs. Later work by Lam *et al.* uses the MDL principle with a GA to evolve BNs [25]. They introduce specialized structure-guided and knowledge-guided reproduction operators, to help make informed decisions about the application of mutation to generate offspring. They found improved performance over the earlier Larrañaga *et al.* paper in the ALARM problem (a standardized problem), showing the value of specialized reproduction operators that adapt to particular problem sets. Both research papers found that the quality of solutions is dependent upon the size of the example set used: more examples usually permit better quality solutions.

Many ideas from work applying evolutionary computation to static BNs extend easily to DBNs. Tucker *et al.* use GAs to evolve DBNs with time lags. These networks permit variables to influence others over more than one time slice or unwinding of the network. Their best results used initial populations that were seeded with link values that were predetermined (via evolutionary programming) to have high correlations, and hence a high chance of being useful in the overall model. In another paper, Tucker *al.* evolve DBNs specialized to model spatial data as denoted by variable influences on a Cartesian coordinate system [41]. This work uses reproduction operators specialized for the spatial nature of the application. Tucker and Liu also look at the evolution of DBNs with changing dependencies [40]. These DBNs are more general-purpose than standard ones, in which variable interdependencies are taken as static over time.

Yu *et al.* infer DBN models of gene expression networks [47]. They incorprate both BDe (Bayesian Dirichlet equivalence) and BIC (Bayesian information criterion) scoring metrics, which are commonly used in BN and DBN learning. They also use a new *influence score* to help improve the relevance of links from recovered networks, and thus reduce false positive (extraneous) links. [46] also use GAs to find DBN structures for analyzing miroarray data, although the details of their method are not given.

### *2.3 Multi-objective problems and Pareto ranking*

A multi-objective optimization problem (MOP) is characterized by a set of multiple objectives or parameters, often of which are related to one another in conflicting, nonlinear ways. Evolutionary computation has been widely applied to MOP's [8][13][44]. Their success in MOP resides in their natural adaptability to the MOP characterization of problems in terms of representation (chromosomes) and performance evaluation (fitness functions), and the correspondence of the multidimensional MOP search space with the schema characterization of evolutionary search [20].

### *2.4 Pareto Ranking*

A popular approach to solving MOP with genetic algorithms is Goldberg's Pareto ranking scheme [15]. The basic idea of a Pareto ranking is to preserve the independence of objectives. This is done by retaining a set of possible solutions, all of which are legitimate solutions with respect to the population at large. This contrasts with a pure genetic algorithm's attempt to ascribe one optimal solution for a MOP, which necessitates a reconciliation of different objective strengths in order to obtain a single optimal solution. The weighted sum used in a single-objective genetic algorithm will also add bias to the kind of result obtained. For many MOP's, relating different objective dimensions with one another can be difficult and arbitrary, and the results are often unsatisfactory.

The following is based on a discussion in [44]. We assume that the MOP is a maximization problem (higher scores are preferred).

Given a problem defined by a vector of objectives $\vec{f} = (f_1, ..., f_k)$ subject to appropriate problem constraints. Consider the optimization problem in which we wish to maximize the $f_i$. Then vector $\vec{u}$ *dominates* $\vec{v}$, or $\vec{u} \succeq \vec{v}$, iff

$$\forall i \in (1, ..., k) : u_i \geq v_i \wedge \exists i \in (1, ..., k) : u_i > v_i.$$

```
Curr_Rank := 1
N := (population size)
m := N
While N ≠ 0 {     /* process entire population */
    For i := 1 to m {     /* find members in current rank */
        If v_i is nondominated {
            rank(v_i) := Curr_Rank
        }
    }
    For i := 1 to m {     /* remove ranked members from population */
        if rank(v_i) = Curr_Rank {
            Remove v_i from population
            N := N-1
        }
    }
    Curr_Rank := Curr_Rank + 1
    m := N
}
```

Fig. 1. Pareto Ranking Algorithm

This implies that a vector is dominated if another vector exists which is better in at least 1 objective, and at least as good in the remaining objectives.

A solution $\vec{v}$ is *Pareto optimal* if there is no other vector $\vec{u}$ in the search space that dominates $\vec{v}$. For a given MOP, the *Pareto optimal set* is the set of vectors $\vec{v}_i$ such that $\forall v_i : \neg\exists\vec{u} : \vec{u} \succeq \vec{v}_i$. For a given MOP, the *Pareto front* is a subset of the Pareto optimal set. A typical MOP will have a multitude of conceivable solutions in its Pareto optimal set. Therefore, in a successful run of a genetic algorithm, the Pareto front will be the set of solutions obtained.

To implement Pareto scoring in a genetic algorithm, chromosome fitness scores take the form of *Pareto ranks*. Figure 1 shows how a Pareto ranking can be computed for a set of vectors. To compute Pareto ranks, the set of nondominated vectors in the population are assigned rank 1. These vectors are removed, and the remaining set of nondominated vectors are assigned rank 2. This is repeated until the entire population is ranked. Genetic evolution then proceeds as usual, using the rank values as reconstituted fitness scores (lower ranks are fitter). Note that Pareto ranks are always relative to the current population. This implies that every generation in a run will have at least a rank 1 set. This has repercussions on performance measurements, as there is no concept of "best solution" amongst all the rank 1 members.

## 3 Experiment

### *3.1 Overview*

We use GA's to evolve DBN's of varying complexity. Our main interest is in evolving the transition network of a DBN. We therefore do not consider the use of the prior network, and simply use the transition network solely during the evaluation of a DBN. This simplification is sensible for a number of reasons. Firstly, the representation is greatly simplified, as the prior network does not need to be denoted nor considered. This in turn simplifies the search space, and hopefully benefits the search for the transition network. When generating test data for target networks, we use a uniform or non-informative prior network, which does not have an influence on the structure of the dynamic network. Therefore, discarding the prior network can be done with no loss in generality of our approach, and permits a more effective analysis of transition network synthesis, which is the more challenging DBN construct to synthesis.

### *3.2 Representation*

The representation scheme used is selected for the intuitive way in which it biases network topology towards sparse structures. Each variable being observed is represented by P sequential genes. For example, if P=4, then a gene would have the form *abcd*, and the entire chromosome for N variables is:

$$a_1 b_1 c_1 d_1 \cdots a_i b_i c_i d_i \cdots a_N b_N c_N d_N$$

Therefore, an N=10 variable network requires a chromosome of 40 genes. A gene represents an inward link from the specified variable, and so there is a maximum of P inward links per variable. Each gene is an integer between 0 and 100. Approximately $100/(N+1)$ integers in this range are interpreted as "skip" codes, and cause that gene to be ignored. Otherwise, considering code $a_i$ above, which denotes a parent link to variable $i$, a value $a_i$ is converted to a variable index $k$ between 1 and N (via modulo N), and represents a parent link from variable $k$ to variable $i$. Duplicate link values in the link fields for a variable are ignored. Hence the skip codes and duplicate codes introduce a natural bias towards sparsely connected networks. In fact, should all P fields for a variable be composed of skip codes, then that variable will not be referenced by the rest of the network.

### *3.3 BIC score*

The nature of probabilistic belief networks is that any arbitrary network structure has the potential to be a correct model for some behavior of interest. However, network structures that denote unnatural relationships of variables will affect the overall complexity of the network, in terms of both the sizes of the conditional probability tables associated with each node, and the structural complexity of the network, for example, the number of links between nodes. Conversely, a network that reflects the actual relationship between variables will have simpler conditional probability tables, as well as a structure more faithful to the phenomenon of interest.

These characteristics of network are accounted for by the the *Bayesian Information Criterion* (BIC) score [37], which is related to the Minimal Description Length (MDL) scoring [35]. The BIC score measures the conditional probability that some data was generated by a given network model. It is an information-theoretic approximation, however, and it does not take into account the prior network influence. This is convenient for us, since we are not considering prior networks in our DBN synthesis. We use a slight variation of the BIC score given in [14] to evaluate the quality of a candidate network in terms of the network's probabilistic likelihood and structural simplicity. As discussed in Section 2.3, the BIC scoring scheme will be applied in a few different ways. The following uses notation based on that in [14], but simplified due to the consideration of the transition network only.

The BIC score uses a heuristic, *Likelihood*, which accounts for the parsimony of the conditional probability tables in each node with respect to the training data. It also incorporates a structural *Complexity*, which evaluates the simplicity of the network in terms of number of links connections:

$$BIC = Likelihood - Complexity$$

Since the *Likelihood* term is larger for more probable networks, it is balanced by subtracting the *Complexity* value from it, since the latter is larger for less-desirable large-sized networks. The definition of both these calculations will now be given.

A training set $S$ has $|S|$ complete observation sequences, which we will use to evolve a DBN that models $S$. One sequence $n_i$ of length $T$ from $S$ has values for the variables:

$$X_i[0], X_i[1], ..., X_i[T]$$

The following notation counts event occurences within sequences:

$$N_{i,j_i,k_i} = \sum_{\ell} \sum_{t} I(X_i[t] = k_i, Parent(X_i[t]) = j_i; x^{\ell})$$

where $I(\phi; x^{\ell})$ is 1 if the event $\phi$ occurs in sequence $x^{\ell}$, and 0 otherwise. The likelihood is then defined as:

$$Likelihood = \sum_{i} \sum_{j_i} \sum_{k_i} N_{i,j_i,k_i} \cdot log(\frac{N_{i,j_i,k_i}}{\sum_{k_i} N_{i,j_i,k_i}})$$

This expression sums for all training sequences ($i$), and possible variable values for nodes ($k_i$) and their parents ($j_i$). This corresponds to the BIC likelihood used in [14].

We use a simple calculation for structural complexity, based on the total link count in the DBN before unwinding. First, the size of the graph $G$ is calculated by finding the number of links in the entire unexpanded network. This is done by summing all the inward links of all nodes (ie. the number of parent nodes):

$$G = \sum_{i} |Parent(X_i)|$$

Then the (structural) complexity is:

$$Complexity = G - N + 1$$

where there are $N$ variables being observed.

### *3.3.1 Single- and multi-objective fitness evaluation*

The above BIC score is used to evaluate network suitability in both single-objective and multi-objective frameworks. We used the BIC formula as given for the single objective fitness function. In such experiments, the goal is to maximixe the BIC score. This is a challenging task, since the natural tendency is to maximize the *Likelihood* term by making a large, maximally connected network. Of course, doing so creates a large *Complexity* term, which is detrimental to an overall good BIC score. Preliminary experiments incorporated a weight $\omega$ into the BIC formula:

$$BIC = Likelihood - \omega \cdot Complexity$$

The weight lets one control the relative balance between likelihood and complexity in the BIC value. However, we found that using a weight is not intuitive,

| *Parameter* | *Value (N=10,20,30)* |
|---|---|
| Runs/experiment | 10 |
| Population size | 500, 1000, 1000 |
| Generations | 60, 100, 100 |
| Chromosome size | N*P |
| Crossover rate | 100% |
| Mutation rate per bit | 1% |
| Tournament size | 5 |

Table 1
Genetic Algorithm Parameters

as it is determined by a complex interaction between the complexity of the target DBN, the number of variables used, and the size of the unwound network (ie. the number of time slices). Hence our single-objective experiments used the unweighted BIC formula.

A multi-objective characterization addresses the above difficulty in balancing the effect of *Likelihood* and *Complexity* in the BIC score. Rather than combine them together in an unnatural, *ad hoc* manner, we instead treat them as separate dimensions of a multi-objective search space, with the intension of maximizing likelihood, while minimizing complexity Using the notion of domination from Section 2.3, we apply a Pareto ranking to the population before evolving a new generation. Hence we do not attempt to numerically balance likelihood and complexity, but instead permit a set of non-dominated DBN solutions that exhibit a range of likelihood and complexity characteristics. When a run terminates, this entire set can be produced as a solution, and the user can determine which balance of likelihood and complexity from the solution DBN's derived is the most suitable.

### *3.4 Other parameters*

Other parameters used by the GA runs are in Table 1. Single values in the table are shared for all networks having total variables N, while multiple values are those that differ for different N. Most of the parameters are standard in the GA literature. Chromosome sizes are calculated by multiplying the total number of variables N by the maximum parent count P, since there are P inward links per variable. Two-point crossover is used. Mutation happens on a gene-by-gene basis, at a probability of 1% per gene. Tournament selection is used, in which 5 random individuals are selected from the population, and the one with the highest fitness is retained as the selected individual for subsequent reproduction.

| Network | *Variables* | *Links* | *Examples* | Time slices | Parent Links (P) |
|---|---|---|---|---|---|
| (a) | 10 | 20 | 10 | 1000 | 4 |
| (b) | 20 | 40 | 200 | 1000 | 4 |
| (c) | 20 | 19 | 200 | 1000 | 3 |
| (d) | 30 | 50 | 300 | 1000 | 4 |

Table 2
Training data

### *3.5 Target networks and training data*

Table 2 summaries training set information about the networks studied. Networks of size 10, 20, and 30 variables were investigated, and the target networks are shown in Figure 2. All variables are connected in the network. The networks are designed to be sparse, and show a variety of connection patterns. The case (c) network is a sparse 20-variable network with topological similarities to "power law" networks [1,23]. It was studied using the multi-objective GA only.

The training data consist of synthetic data created with the "Bayes Net Toolbox" for Matlab [28]. Sequence sizes are 1000 time slices, and variables take one of 3 values in the sequences.

Since the target networks in Figure 2 are canonical solutions, there is no need for testing data to cross-validate evolved solutions. Any DBN configuration can be used to model the training data. Differences between configurations are evident in their likelihood and structural simplicity, as encoded by the BIC evaluation. Since we are primarily interested in the structural accuracy between the evolved network and corresponding target network, simply reporting the structural accuracy of evolved results is sufficient for evaluating the effectiveness of evolved solutions.

## 4 Results

Some description of how the result data was tabulated is necessary. Single-objective GA runs are easily tabulated, since the single network with the best fitness score is designated as the solution for the run. Multi-objective runs using Pareto rankings are more difficult to evaluate. The end result of a run is a set of solutions belonging to the rank 1 Pareto set for that run. Therefore, a variety of solutions are given, which are all non-dominated within the final population. These networks have a variety of likelihood and structural complexity measurements. To evaluate the multi-objective experiments, the rank

| Network & N | Single objective | | | | Multi-objective: Pareto set S from 10 runs | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Avg | Best | | | | Like. | Median | Struct. | | | Best match | | |
| | E | E | FP | FN | \|S\| | E | E | E | FP | FN | E | FP | FN |
| (a) 10 | 7 | 2 | 1 | 1 | 6 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| (b) 20 | 22 | 18 | 9 | 9 | 7 | 10 | 7 | 10 | 5 | 5 | 6 | 4 | 2 |
| (c,3) 20 | 25 | 21 | 11 | 10 | 14 | 15 | 7 | 3 | 2 | 1 | 2 | 2 | 0 |
| (c,4) 20 | - | - | - | - | 14 | 24 | 16 | 12 | 11 | 1 | 11 | 11 | 0 |
| (d) 30 | 32 | 24 | 12 | 12 | 14 | 23 | 20 | 18 | 9 | 9 | 18 | 9 | 9 |

Table 3
Results summary: All values are number of structural errors (E) compared to target network.

1 set from each run was incorporated into one large solution population. Duplicates were removed. Then the networks were re-ranked with Pareto ranking, to determine the overall rank 1 set for all 10 runs.

When analyzing the multi-objective runs, it is scientifically invalid to chose an overall best solution from the rank 1 set by comparing the structure of these networks with that of the target network. This is because there is no inherent factor favouring any of these rank 1 solutions over any other in the set. Some have stronger likelihood scores, while others are structurally simpler. Potentially any may be the closest match to the target network, whose structure is of course unknown to the genetic algorithm. Hence the performance of these runs must be evaluated by considering the range of values obtained in the universal rank 1 set for the runs. Of course, this selection issue is irrelevant for the single objective GA, since one single network is identified as the solution at the end of each run.

Table 3 summarizes the performance of the experiments. The *Network* column shows the target network from Figure 2, along with the associated number of variables $N$. In the case of network (c), experiments using inward parent link gene sizes (P) of 3 or 4 are given in (c,3) and (c,4) respectively. In the single objective columns, the average error is computed by taking the single highest-scoring solution per run, comparing its structural differences with the target network, and finding the average errors for all such networks for the 10 runs. *Best* specifies the network with the highest fitness score from all 10 runs. *FP* and *FN* are the counts for extraneous links (false positive) or missing links (false negative) respectively. The multi-objective portion is calculated from the rank 1 set for all 10 runs, as described earlier. $|S|$ is size of the rank 1 set. Since this set denotes a variety of solutions, we show the number of structural errors for the networks at the extreme ends of the multi-dimensional space – the ones with the highest likelihood and structural complexity scores. We also

show the performance of the median network falling between these extrema. Since we are trying to create sparse networks, we also include the false positive and false negative scores for the network with the best structural complexity score. The *best match* columns show the network from the rank 1 set that most closely matches the target network.

Some trends can be seen in the results. Naturally, the larger target networks are more challenging than smaller ones. In the single-objective runs, the best solution per experiment (the network with the highest fitness score) also had the lowest structural error count of all solutions. The multi-objective results are superior in all instances to the single-objective weighted scoring runs. The network with the highest likelihood score in the multi-objective runs tended to have worse performance with respect to structural errors. This is because such networks use more links to increase their likelihood scores, at the expense of structural complexity. On the other hand, the runs for networks (b) and (c) show that the networks with the best structural scores (lower link counts) are not always the best performers from the rank 1 set. In such cases, those networks lack required links, as is seen by their higher FN values when compared to the best match network. The best performing networks were closer to the median networks in these cases.

Our structural complexity score (the total number of links in a network) tends to result in networks that minimize the number of links used. For most experiments (except network (c)), for every extra link used in the FP column, there is a valid link missing in the FN. Note, however, that there are far fewer links in the networks than are potentially possible, since the gene sizes ranged from 40 to 120 links for the different networks. Hence the structural complexity score was effective in simplifying networks in both single and multi-objective experiments.

The 20-variable power law network (c,3) using the 3-parent chromosome performed much better in the multi-objective runs, than the corresponding single-objective result – even moreso than the larger 20-variable network (b) in the single-objective experiment. To test the effect of chromosome size, we ran a new set of multi-objective runs on the power law (c) network using the 4-parent chromosome of the other runs. The results are shown on line (c,4). The performance was considerably worse, but still better than the single-objective run with 3-genes in (c,3). This shows that the chromosome bias is an important and effective influence on derived network structure.

Figure 3 shows the best DBN obtained for the 20-variable network (b) in Figure 2, obtained with the multi-objective experiment. This network has 6 structural errors from the target network – 4 false positive links (bold arrows) and 2 false negative (dashed arrows). Interestingly, both false negative links are reflections of the false positive links. Thus a correlation was discovered

between these two pairs of variables.

## 5 Discussion

A number of preliminary experiments were done to examine possible representations for the DBN. The first representation tried was an adjacency matrix. Here, the boolean entries in a K by K matrix denote the existence of links between the K variables. Unfortunately, this representation naturally promotes highly connected networks, and often maximally connected ones, since such networks have high probability scores as determined by the BIC heuristic. Even the introduction of harsh structural complexity penalties could not prevent the tendency for evolving overly connected networks.

We also tried using a cellular encoding language with genetic programming to construct network graphs [16]. Cellular encoding has been found effective for evolving graph structures to be used as finite automata and artificial neural networks. Cellular operators generate a complex graph using operations that piecewise generate and manipulate components from simpler graph instances. Unfortunately, our trials with cellular GP were unsuccessful. The sizes of cellular GP trees were linearly proportional to the size of target DBN graphs, since GP expressions were virtually 1:1 mappable to links and nodes in the target DBN graph. The reason for this failure was because our desired networks were asymmetric, and lacked symmetries and patterns that are usually exploited by cellular encodings. Should our DNB's have had structural regularities, cellular encoding might have been more successful.

Our adoption of multi-objective Pareto ranking was in reaction to experiences with early experiments that used the weighted sum in the BIC scoring. Different networks required their own *ad hoc* tuning of weights to get acceptable results. The use of Pareto ranking for likelihood-structural complexity space was a considerable benefit in this regard, as it prevented our supplying weight parameters, and also produced good results. It should be mentioned, however, that a fortuitous selection of weights can easily result in much better performance, should those weights result in a search space more amenable to navigation by the genetic algorithm. Our experience, however, is that the effort required to find helpful weight parameters overshadows any benefits that might arise with them.

Tucker *et al.* use a GA to evolve DBNs [42]. They study DBNs with time lags, in which variables can influence other variables up to a finite number of time slices away. Such DBNs are more general than the DBN normally studied, in which all links span single time steps. Their chromosomal representation for DBNs uses one gene per parent-child link, with an additional value specifying

the time lag spanned by that link. Additionally, they automatically include self-referential links for each variable, spanning 1 time slice. The rationale for this is that these links are common in some application, such as the chemical process application that they were modeling. Their representation therefore permits fairly sparse networks when the chromosome size is appropriately fixed. The automatic creation of self-referential links does bias the form of networks, and can be detrimental for modeling other phenomena. They incorporate a BIC scoring scheme (unweighted sum of likelihood and structural penalty), and compare it with a log likelihood heuristic. They use evolutionary programming to determine a set of initial chromosomes to use for seeding a population given to a GA. The GA uses knowledge-guided reproduction operators, to help make processing sensitive to the example data being used. Since their problem data and type of DBN is different than ours, it is difficult to compare our performance with theirs. Their EP-seeded-GA experiments for N=10 and N=20 variables, with 2000 fitness evaluations (function calls), report structural error rates of 9.2 and 20.6 respectively.

Our chromosome representation is similar to the one outlined in Yu *et al.*, as their chromosome consists of a set of parent nodes per node [47]. It is not clear whether these parent sets are of fixed length. They also use the BIC scoring, presumably with the unweighted composition of likelihood and structural complexity metrics, as we use with the single-objective GA. They also introduce an influence score metric, which determines the strength of causality for links in the network. Their intention is to simulate gene expression networks, and so their test networks are very sparsely connected. Their test networks have 20 nodes, but only 8-12 of the nodes are causally related. The remaining nodes are distraction noise. They also use between 25 to 5000 time slices, and 100 examples per network. The 25 time slice experiments report FN and FP rates of about 60%, while the experiments using 2000 and 5000 slices have FN and FP rates of approximately 3%. Since they are primarily motivated to simulate biological networks, they are strongly motivated to learn DBN's using smaller numbers of time slices typical with such applications (25 to 100). Hence the precision of resulting networks is adversely affected to a significant degree when using these smaller time sequences.

## 6 Conclusion

There are a number of ways this research can be extended. The use of local search to refine network structures is worth consideration. Yu *et al.* use an influence heuristic to evaluate the strength of correlation for linkages [47]. They suggest that false positive links are often useful, as they may denote relationships between a node and its grandparents or sibling, rather than from its intended parent. Such effects were also seen in by Tucker *et al.* [42], who

term them "spurious correlations". Perhaps a simple local search strategy of moving links from parents to nearby ancestors, or inverting the directions of links, would correct many structural errors.

Instead of or in addition to local search, the initial population could be seeded with links that are determined to be strong, as is done in [42]. They found that GA populations seeded with strong links performed much better than GA's with non-seeded populations. The latter spent a lot of search trying to find useful variable correlations, which was detrimental to the overall search for a global DBN.

Acknowledgement: This research is supported by NSERC Operating Grant 138467-1998 and an NSERC USRA.

## References

[1] A.-L. Barabasi and R. Albert. Emergence of Scaling in Random Networks. *Science*, 286(15):509–512, October 1999.

[2] R.R. Bouckaert. Probabilistic Network Construction Using the Minimum Description Length Principle. Technical Report UU-CS-1994-27, Utrecht University, Dept of Computer Science, July 1994.

[3] W.L. Buntine. Operations for Learning with Graphical Models. *Journal of Artificial Intelligence Research*, 2:159–225, 1994.

[4] E. Castillo, J.M. Gutierrez, and A.S. Hadi. *Expert Systems and Probabilistic Network Models*. Springer Verlag, 1997.

[5] E. Charniak. Bayesian Networks without Tears. *AI Magazine*, pages 50–63, Winter 1991.

[6] D.M. Chickering. Learning Bayesian Networks is NP-Complete. In D. Fisher and H.-J. Lenz, editors, *Learning from Data: AI and Statistics*. Springer Verlag, 1996.

[7] L. Chrisman. A Roadmap to Research on Bayesian Networks and other Decomposable Probabilistic Models. citeseer.nj.nec.com/ chrisman98roadmap.html. Last accessed April 1, 2005.

[8] C.A. Coello Coello, D.A. Van Veldhuizen, and G.B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, 2002.

[9] G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[10] C. Cotta and J. Murzabal. Towards a More Efficient Evolutionary Induction of Bayesian Networks. In *Proceedings PPSN 2002*, pages 730–739, 2002.

[11] S. Davies and A. Moore. Bayesian Networks for Lossless Dataset Compression. *Proceedings of the 5th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 387–391, 1999.

[12] L.M. de Campos and J.F. Huete. Approximating Causal Orderings for Bayesian Networks using Genetic Algorithms and Simulated Annealing. Technical Report #DECSAI-99021, U. of Grenada, Uncertainty Treatment in Artificial Intelligence Group, May 1999.

[13] C.M. Fonseca and P.J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[14] N. Friedman, K. Murphy, and S. Russell. Learning the Structure of Dynamic Probabilistic Networks. In *Proc. Uncertainty in AI (UAI'98)*. Morgan Kaufman, 1998.

[15] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley, 1989.

[16] F. Gruau. Genetic Micro Programming of Neural Networks. In K.E. Kinnear, editor, *Advances in Genetic Programming*, pages 495–518. MIT Press, 1994.

[17] H. Guo and W. Hsu. A Survey of Algorithms for Real-Time Bayesian Network Inference. In *Proc. Joint AAAI-02/KDD-02/UAI-02 Workshop on Realtime Decision Support and Diagnosis Systems*, 2002.

[18] S. Harwood and R. Scheines. Genetic Algorithm Search over Causal Models. Technical Report CMU-PHIL-131, Carnegie Melon University, Dept. of Philosophy, 2002.

[19] D. Heckerman. A Tutorial on Learning with Bayesian Networks. Technical Report MSR-TR-95-06, Microsoft Research, March 1995.

[20] J.H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 1992.

[21] W.H. Hsu, H. Guo, B.B. Perry, and J.A. Stilson. A Permutation Genetic Algorithm For Variable Ordering In Learning Bayesian Networks From Data. In W.B. Langdon *et al.*, editor, *Proc. GECCO 2002*, pages 383–390. Morgan Kaufmann, 2002.

[22] D. Husmeier. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics*, 19(17):2271–2282, 2003.

[23] H. Jeong, B. Tombor, R. Albert, Z.N. Oltvai, and A.-L. Barabasi. The large-scale organization of metabolic networks. *Nature*, 407:651–654, October 2000.

[24] S.Y. Kim, S. Imoto, and S. Miyano. Inferring gene networks from time series microarray data using dynamic Bayesian networks. *Briefings in Bioinformatics*, 4(3):228–235, 2003.

[25] W. Lam, M.L. Wong, K.S. Leung, and P.S. Ngan. Discovering Probabilistic Knowledge from Databases Using Evolutionary Computation and Minimum Description Length Principle. In J.R. Koza *et al*, editor, *Proc. Genetic Programming 1998*, pages 786–794. Morgan Kaufmann, 1998.

[26] P. Larranaga, M. Poza, Y. Yurramendi, R.H. Murga, and C.M.H. Kuijpers. Structure Learning of Bayesian Networks by Genetic Algorithms: A Performance Analysis of Control Parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):912–926, 1996.

[27] F. Markowetz. A bibliography on learning causal networks of gene interactions. http://www.molgen.mpg.de/~ markowet/docs/network-bib.pdf. Last accessed April 1, 2005.

[28] K.P. Murphy. Bayes Net Toolbox for Matlab. www.cs.ubc.ca/~ murphyk/Software/BNT/bnt.html. Last accessed March 16, 2005.

[29] K.P. Murphy. Dynamic Bayesian Networks. www.cs.ubc.ca/~ murphyk/Papers/ dbnchapter.pdf. To appear in *Probabilistic Graphical Models*, M. Jordan. Last accessed April 13, 2005.

[30] J.W. Myers, K.B. Laskey, and K.A. DeJong. Learning Bayesian Networks from Incomplete Data using Evolutionary Algorithms. In W. Banzhaf *et al*, editor, *Proc. GECCO-99*, pages 458–465, 1999.

[31] A.V. Nefian, L. Liang, X. Pi, X. Liu, and K. Murphy. Dynamic Bayesian Networks for Audio-Visual Speech Recognition. *EURASIP Journal on Applied Signal Processing*, pages 1–15, 2002.

[32] J. Pearl. *Causality*. Cambridge, 2000.

[33] M. Pelikan, D.E. Goldberg, and E. Cantu-Paz. BOA: the Bayesian Optimization Algorithm. In W. Banzhaf *et al*, editor, *Proc. GECCO-99*, pages 525–532, 1999.

[34] B.-E. Perrin, L. Ralaivola, A. Mazurie, S. Bottani, J. Mallet, and F. d'Alche Buc. Gene networks inference using dynamic Bayesian networks. *Bioinformatics*, 19:ii138–ii148, 2003. Supplement 2.

[35] J. Rissanen. Stochastic complexity and modeling. *The Annals of Statistics*, 14(3):1080–1100, 1986.

[36] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2 edition, 2003.

[37] G. Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2):461–464, 1978.

[38] B. Sierra and P. Larranaga. Predicting the Survival in Malignant Skin Melanoma Using Bayesian Networks Automatically Induced by Genetic Algorithms. An Empirical Comparison between Different Approaches. *Artificial Intelligence in Medicine*, 14(1–2):215–230, 1998.

[39] T.A. Stephenson. An Introduction to Bayesian Network Theory and Usage. Technical Report 00-03, IDIAP, February 2000.

[40] A. Tucker and X. Liu. Learning Dynamic Bayesian Networks from Multivariate Time Series with Changing Dependencies. In *Proc. 5th Intelligent Data Analysis Conference (IDA 2003)*, pages 100–110. Springer-Verlag, 2003. LNCS 2810.

[41] A. Tucker, X. Liu, and D. Garway-Heath. Spatial Operators for Evolving Dynamic Bayesian Networks from Spatio-temporal Data. In E. Cantu-Paz *et al.*, editor, *Proc. GECCO 2003*, pages 2360–2371. Springer-Verlag, 2003.

[42] A. Tucker, X. Lui, and A. Ogden-Swift. Evolutionary Learning of Dynamic Probabilistic Models with Large Time Lags. *International Journal of Intelligent Systems*, 16(5):621–646, 2001.

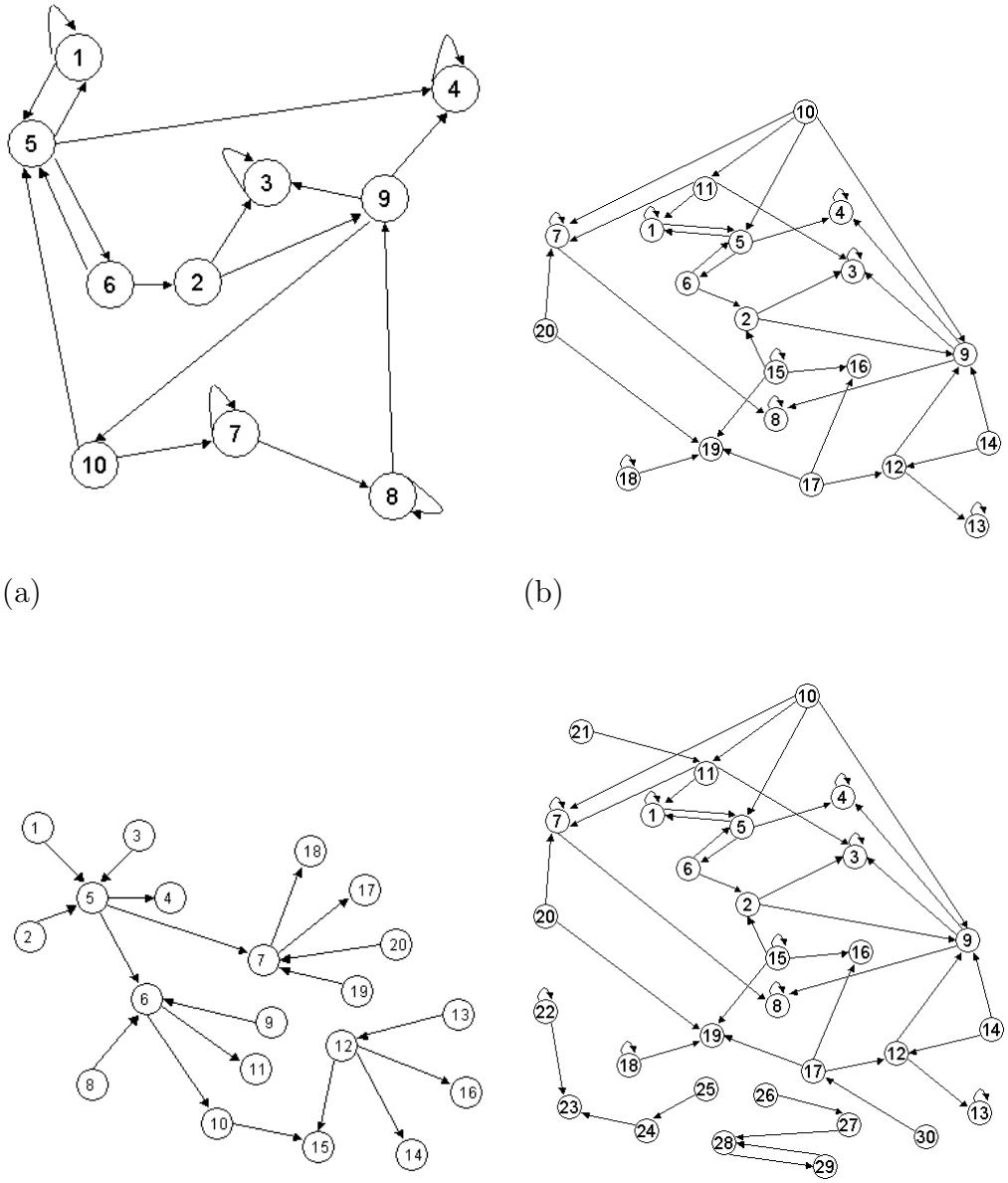
[43] S. van Dijk, D. Thierens, and L.C. van der Gaag. Building a GA from Design Principles for Learning Bayesian Networks. In E. Cantu-Paz *et al.*, editor, *Proc. GECCO 2003*, pages 886–897. Springer-Verlag, 2003.

[44] D.A. van Veldhuizen and G.B. Lamont. Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. *Evolutionary Computation*, 8(2):125–147, 2000.

[45] M.L. Wong, S.Y. Lee, and K.S. Leung. A Hybrid Data Mining Approach to Discover Bayesian Networks Using Evolutionary Programming. In W.B. Langdon *et al.*, editor, *Proc. GECCO 2002*, pages 214–221, 2002.

[46] C.C. Wu, H.C. Huang, H.F. Juan, and S.T. Chen. GeneNetwork: an interactive tool for reconstruction of genetic networks using microarray data. *Bioinformatics*, 20(18):3691–3693, July 2004.

[47] J. Yu, V.A. Smith, P.P. Wang, A.J. Hartemink, and E.D. Jarvis. Advances to Bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*, 20(18):3594–3603, July 2004.




(a) (b)

(c) (d)

Fig. 2. Target networks

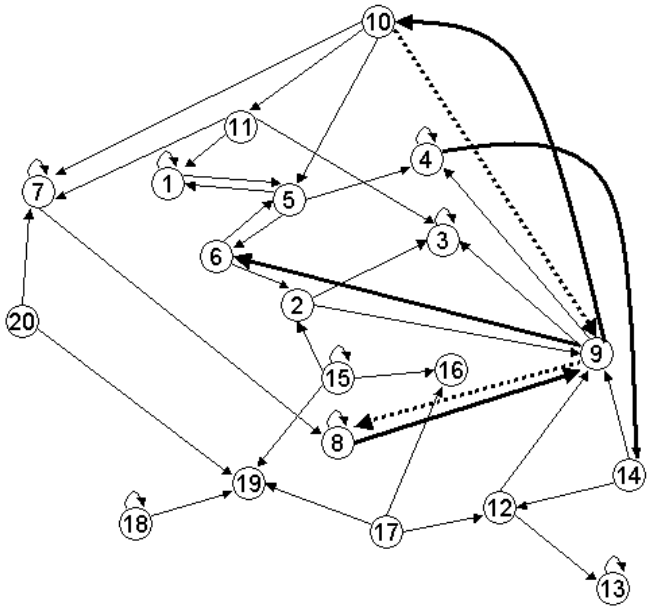



Fig. 3. Best network for network (b) from multi-objective runs. Bold links are erroneous (false positive) and dashed links are missing (false negative).