**Proceedings of the 2007 IEEE Symposium on
Computational Intelligence and Data Mining (CIDM 2007)**

K2GA: Heuristically Guided Evolution of Bayesian Network Structures from Data

Eli Faulkner
Quantum Leap Innovations
3 Innovation Way, Suite 100
Newark DE, 19711
etf@quantumleap.us

***Abstract*— We present K2GA, an algorithm for learning Bayesian network structures from data. K2GA uses a genetic algorithm to perform stochastic search, while employing a modified version of the K2 heuristic to score proposed networks and improve future generations. We show each component of K2GA, a combination of these components to form the basic algorithm, extensions to the algorithm for improved accuracy, and numerical results.**

I. Introduction

A Bayesian Belief Network (BBN) is a structure which efficiently encodes a joint probability distribution, and allows probabilistic inferences to be made among the variables which it encodes. During the 1980's and 1990's, many researchers developed a large number of algorithms intended on allowing decision makers to use Bayesian Belief Networks as a tool. Developing BBNs manually is often impractical and inherently not scalable. To make tools which used BBNs practical, techniques to learn Bayesian networks from data needed to be developed. While great advances have been made on this problem, it has also been shown to be NP hard and improvements are still being made.

Two main approaches have been developed to efficiently learn the best Bayesian network structures given a data set. One class of approaches are the constraint based methods [1] developed at Carnegie Mellon University. Constraint based methods learn conditional independence relationships among the variables, and use these independence's to determine the network structure. The second class of approaches are the model selection approaches [2], [3] developed at Stanford University. Model selection methods are optimization based methods which search the space of causal structures and determine the structures ability to describe the data set as a model score. Model selection methods can be generally broken down into stochastic search methods and heuristic search methods.

We present K2GA; a model selection method which provides several advantages over current techniques by a novel combination of genetic algorithms and the K2 heuristic.

Section II provides necessary background knowledge and definitions on Genetic algorithms, Bayesian Networks, and Bayesian network structure learning. Section III shows the K2GA algorithm and each of its components. Section IV shows extensions and improvements to the basic algorithm. Section V gives numerical results for the algorithm. Section VI compares K2GA to other existing model selection methods.

This work was funded in part by the United States Office of Naval Research under Contract N00014-02-C-0320

II. Background

The work presented in this paper uses ideas from Genetic Algorithms, Bayesian Belief Networks, and Bayesian Belief Network learning algorithms.

A. Genetic Algorithms

In this work we use an elitist genetic algorithm. The genetic algorithm framework is straightforward and similar to those found in undergraduate textbooks [4]. Our genetic algorithm implementation works by defining four components.

- A representation for members of the population. Each member represents one point in our search space.
- A crossover operator for taking two members of a population as parents and creating a new member, or child, which inherits traits from its parents.
- A mutation operator for making small changes to a member of the population.
- A fitness function for evaluating the quality of each member in the population.

Using these components, we can find optimal members by following the loop shown in Figure 1. We begin by creating an initial population of some predefined number of members. We then enter the cycle by scoring each member using the fitness function. When each member is scored we allow some percentage of the highest scoring members to enter the next population. From these surviving members, we will randomly pick pairs of members and perform the crossover operation on them until the culled population is returned to its original size. Finally, we apply the mutation operation to invoke local changes on some percentage of the new population. At this point we can return to the scoring stage.

Each cycle through this genetic algorithm is called a generation. As we run through the generations the members which represent the top scoring points in the search space are recorded. After some number of cycles we return the top scoring member or members which were found during the search.

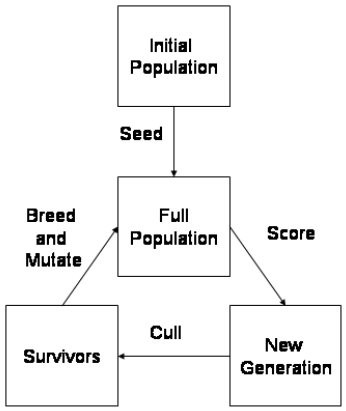


Fig. 1. A Genetic Algorithm Main Loop

B. Bayesian Belief Networks

We will forgo detailed definitions of BBNs and recommend that interested readers familiarize themselves with the work of Pearl [5] and of Sprites, Gylmour and, and Scheines [1]. We will, however, give some basic definitions which will assist the reader.

Definition Given a set of random variables $\mathbf{X} = \{X_1, ..., X_N\}$, a **Bayesian Belief Network** is a structure which efficiently encodes the joint distribution by exploiting conditional independence's among the variables. A BBN is represented by $\mathcal{B} = (\mathbb{G}, \Theta)$, where $\mathbb{G}$ is a Directed Acyclic Graph (DAG), and $\Theta = \{\Theta_1, ..., \Theta_N\}$ represents conditional probabilities for $P(X_i|\Pi_i)$, where Π_i are the parents of X_i in $\mathbb{G}$. Using this representation we have $P(X_1, ..., X_n) = \prod_{i=1}^{N} P(X_i|\Pi_i)$

Throughout this paper we will assume that each variable in our BBN is discrete. We will also assume the following definition.

Definition A **Topological Ordering** $\prec$ of a set of nodes $\{X_i\}_{i=1}^{N}$ in a network is an ordering such that $\forall i, j$, if $X_j \in \text{Anc}(X_i), X_j \prec X_i$, where $\text{Anc}(X)$ are the ancestors of X.

C. Learning Bayesian Network Structures from Data

The work described in this paper is a model selection method grounded in the work of Cooper and Herskovits [2], and of Heckerman [3]. Given a data set D containing cases describing the variables $X_1, ..., X_N$, we wish to find a DAG structure which most likely represents the true conditional independence relationships among the variables.

In a model selection approach to structure learning we will search the space of all DAGS and score each DAG against the data set D using a fitness measure. The most commonly used fitness measure [2], [3], which can be interpreted as the probability of the data set D given some proposed model structure $\mathbb{G}$, is shown in (1) with its parameters shown in Table I.

r_i	The number of states of variable X_i
q_i	The number of potential parent configurations of X_i
N_{ijk}	The number of cases in D where X_i is in its j^{th} state and its parents are in their k^{th} state
N_ij	$\sum_{k=1}^{r_i} N_{ijk}$
S	An equivalent sample size
a_{ijk}	$\frac{S}{r_i q_i}$
a_{ij}	$\frac{S}{q_i}$

TABLE I
THE PARAMETERS TO THE BAYESIAN SCORING FUNCTION SHOWN IN (1)

$$P(D|\mathbb{G}) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(a_{ij})}{\Gamma(a_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(a_{ijk} + N_{ijk})}{\Gamma(a_{ijk})} \quad (1)$$

An important feature of equation 1 is the fact that if we take the log of the score, then the equation reduces to the sum of the local score at each node. This leads to the concept of a decomposable scoring function [6].

Definition A scoring criteria S for a DAG $\mathbb{G}$ is **decomposable** iff $S(\mathbb{G}) = \sum_{i=1}^{N} s(X_i, \Pi_i)$. Each term in this sum will be referred to as the **local score** of the node. Each node and its parents will be referred to as a **local structure**.

There have been several model selection methods which approached the problem of learning a Bayesian Network structure from data. Methods relevant to this paper include the K2 heuristic and other genetic algorithm based searches.

1) The K2 Heuristic : The K2 Heuristic described in [2] takes a data set D and a node ordering $\prec$, and repeatedly attempts to add edges oriented in the direction defined by $\prec$ which will increase the Bayes score. The K2 algorithm is shown in Algorithm 1.

Algorithm 1 The K2 Algorithm

Require: $N > 0$, $\prec$ an ordering of the nodes, D a data set, M the maximum indegree of a node
  for $i = 1$ to N **do**
    $\pi_i = \emptyset$
    $P_{old} = score(D, X_i, \pi_i)$
    oktoProceed = true
    while okToProceed && $|\pi_i| < M$ **do**
      let X_j be the node such that $X_j \prec X_i$, and $X_j \notin \pi_i$ that maximizes $score(D, X_i, \pi_i \cup X_j)$
      $P_{new} = score(D, X_i, \pi_i \cup X_j)$
      if $P_{new} > P_{old}$ **then**
        $P_{old} = P_{new}$
        $\pi_i = \pi_i \cup X_j$
      else
        okToProceed = false
      end if
    end while
  end for

We can see that the K2 Algorithm builds the network score as the sum of local scores. The K2 algorithm also introduces another common model selection parameter M, the maximum number of parents which a node can have. Limiting the number of parents for each node is a typical approach in modeling with BBNs. The K2 Algorithm is very powerful if one can determine the topological ordering of the nodes. Unfortunately, this is not usually possible.

2) Genetic Algorithm Search: To alleviate the problem of needing to know the variable ordering in K2, one could apply a search technique to the space of variable orderings and run the K2 algorithm repeatedly on each ordering. This technique was described by Hsu et al. [7].

An alternative approach to model selection using genetic algorithms is given by Larrañaga et al. [8]. They present two approaches to model selection. Both approaches search the space of adjacency matrices. Their first approach assumes a fixed ordering of the nodes to ensure that each proposed DAG is acyclic. As in the case of the K2 algorithm, the proper node ordering can not always be found. Their second approach searches directed graphs, and uses a repair operator when a directed graph with a cycle is encountered.

III. K2GA

We now present the K2GA algorithm. K2GA is a genetic algorithm search which uses a modified version of the K2 heuristic to improve search efficiency. In this section we present the four components needed to define the genetic algorithm: member representation, a crossover operator, a mutation operator, and a fitness function.

A. Member Representation

A member in K2GA is a representation of a DAG structure.

We will use the following representation for DAG Structures. Let $X = \{X_1, ..., X_N\}$ be a set of variables. Let $\Theta = \{\Theta_1, ..., \Theta_N\}$ be a set of real numbers such that $\forall i, \Theta_i \in [0, 1]$ and $\forall i, j\ \Theta_i \neq \Theta_\text{J}$. Let $\mathbb{B} \in \{0, 1\}^{N \times N}$ be a binary adjacency matrix such that $\mathbb{B}_{ij} = 1$ iff X_i and X_j have a parent/child relationship in either order in the DAG. Note that $\mathbb{B}$ is symmetric.

Using the definition of topological ordering, and $\prec$ to define that ordering, we let $\Theta_i < \Theta_j$ iff $X_i \prec X_j$. Then using Θ and $\mathbb{B}$, we can define a DAG as $\mathbb{G} = (\mathbb{B}, \Theta)$. Then for $\mathbb{G} = (\mathbb{B}, \Theta)$, X_i is a parent of X_j iff $\Theta_i < \Theta_j$ and $\mathbb{B}_{ij} = 1$.

Using $\mathbb{G} = (\mathbb{B}, \Theta)$ we can represent any DAG structure, and $\forall(\mathbb{B}, \Theta)$ it must be true that $\mathbb{G} = (\mathbb{B}, \Theta)$ represents a correct DAG structure with no cycles.

B. Crossover

Breeding two members $\mathbb{G}^{(0)}$ and $\mathbb{G}^{(1)}$ to create a child $\mathbb{G}^{(c)}$ involves breeding the Θ's and the $\mathbb{B}$'s. Since we are breeding members which have been scored, we can use the score of graph $\mathbb{G}$, call it $\sigma(\mathbb{G})$, in this operation.

Since the Θ vectors have been normalized, and are on the same scale, we can compute the Θ vector for the child as the weighted average of the parents Θ vectors.

$$\Theta_i^{(c)} = \frac{\sigma(\mathbb{G}^{(0)})\Theta_i^{(0)} + \sigma(\mathbb{G}^{(1)})\Theta_i^{(1)}}{\sigma(\mathbb{G}^{(0)}) + \sigma(\mathbb{G}^{(1)})} \tag{2}$$

To compute $\mathbb{B}^{(c)}$, we will let $p = \frac{\sigma_i(\mathbb{G}^{(0)})}{\sigma_i(\mathbb{G}^{(0)}) + \sigma_i(\mathbb{G}^{(1)})}$ and use Equation (3) to determine $\mathbb{B}^{(c)}$, where $\mathbb{B}_i$ is the i^{th} row of the adjacency matrix. We then must fill $\mathbb{B}$ to symmetry.

$$\mathbb{B}_i^{(c)} = \begin{cases} \mathbb{B}_i^{(0)} & \text{if } random(0,1) \leq p \\ \mathbb{B}_i^{(1)} & \text{if } random(0,1) > p \end{cases} \tag{3}$$

During the execution of Algorithm 5 we can create the $breedMembers(P, size)$ routine by repeatedly selecting 2 members from the initial population, and applying Equations (2) and (3) to create children until the size of the initial population and the new children equals the $size$ parameter.

C. Mutation

There are 2 steps to mutating a DAG $\mathbb{G} = (\mathbb{B}, \Theta)$. First we mutate Θ, then we mutate $\mathbb{B}$.

The mutation operation in K2GA has a very important and interesting parameter, the connection probability matrix. The connection probability matrix is an $N \times N$ matrix C where C_{ij} represents the belief that X_j is a good parent/child connection for X_i. We restrict C such that $\sum_{j=1}^{N} C_{ij} = 1$.

The most basic C which one could use is shown in (4). In this case each node has an equal probability of undirected connection to each other node.

$$C_{ij} = \begin{cases} \frac{1}{N-1} & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \tag{4}$$

We will use this C to perform mutation on $\mathbb{B}$ through a weighted roulette wheel approach. Using our connection probability matrix C, where C_{ij} is a probability that X_j will be added to $\mathbb{B}$ when we are connecting nodes to nodes X_i, Algorithm 2 can be used to make local changes to $\mathbb{B}$. As we will see when discussing the scoring function, our mutation operation only needs to add edges to $\mathbb{B}$.

Algorithm 2 The mutate operator for $\mathbb{B}$

Require: $\mathbb{B}$ the initial adjacency matrix, and C the connection probability matrix, and $maxNewConnections$ the maximum number of connections that any node can have.
for $i = 1$ to N **do**
  numConnections = randomInt(0, maxNewConnections)
  for $k = 1$ to $numConnections$ and $\mathbb{B}$ contains at least one 0 in a non-diagonal **do**
    pick a j using weighted roulette wheel and ignoring all j where $C_{ij} = 1$.
    $\mathbb{B}_{ij} = \mathbb{B}_{ji} = 1$
  end for
end for

Our mutate operator for Θ is easy, we will make local changes in each Θ_i, as shown in Algorithm 3.

Algorithm 3 The mutate operator for Θ

Require: $L \in (0, 1)$ a bound on the magnitude of the move.
  for all $\Theta_i \in \Theta$ **do**
    $s = \frac{L}{|\Theta|}$
    $\Theta_i = \Theta_i + \text{random}(-.5, .5) * s$
  end for

This function will make local changes in each Θ_i, therefore changing the topological order of the variables when they cross values.

Applying these operations guarantees that we can reach any DAG as long as $\forall i, j \leq N, C_{ij} \neq 0$. If we wish to restrict some nodes X_i and X_j from being connected (either by preprocessing or by user input), then we simply make $C_{ij} = 0$ and ensure that no member of the initial population has that connection.

During the execution of Algorithm 5 we can create the $mutateMembers(P, prob, C)$ routine by selecting each member of the population with probability $mutateProbability$, and applying Algorithms 2 and 3 to the selected members.

D. Fitness

The final component to our genetic algorithm is the fitness function. To compute the fitness of a member we will use a modified version of the K2 algorithm. Given a node ordering, the K2 algorithm [2] finds a DAG which maximizes the score in Equation 1 by attempting to add edges in the direction given by the ordering and only keeping edges which cause a increase in the score. Our modified K2 algorithm, shown in Algorithm 4, begins with a DAG pattern and applies the K2 process to find an optimal sub-DAG while only attempting to add edges which exist in the original DAG. Thus the result of the algorithm is a DAG which contains a subset of the edges from the original DAG, and a score for this new sub-DAG.

Using the modified K2 Algorithm to score members, a member can modify its state to increase its score. This presents an interesting twist over other algorithms. After scoring, the members new "self-optimized" representation is the one used in future generations!

E. K2GA as a Genetic Algorithm

Using the components described above, we use Algorithm 5 as our final model selection algorithm. This algorithm describes the GA process shown in Figure 1.

IV. K2GA Enhancements

Beyond the basic K2GA algorithm, there are some optional enhancements to decrease solve time or increase model quality.

A. Local Score Caching

Due to the modified K2 algorithm's use of a decomposable scoring function, model scores are sums of local scores. Local structures which produce high scores will occur frequently in our search, especially in later generations. In our implementation we keep a cache of local scores. At the end of each generation, we can pass through the cache and remove cached scored which did not occur frequently. Reading the scores from a cache is much faster than computing Equation 1, since we need to pass through the data each time we compute this score.

Algorithm 4 The Modified K2 Algorithm

Require: $N > 0$, $\prec$ an ordering of the nodes, $\mathbb{B}$ an adjacency matrix such that $\mathbb{B}_{i,j} = 1$ iff X_i can be connected to X_j, D a dataset, M the maximum indegree of a node
  for $i = 1$ to N **do**
    $\pi_i = \emptyset$
    $P_{old} = score(D, X_i, \pi_i)$
    oktoProceed = true
    while okToProceed && $|\pi_i| < M$ **do**
      let X_j be the node such that $\mathbb{B}_{i,j} = 1$, $X_j \prec X_i$, and $X_j \notin \pi_i$ that maximizes $score(D, X_i, \pi_i \cup X_j)$
      $P_{new} = score(D, X_i, \pi_i \cup X_j)$
      if $P_{new} > P_{old}$ **then**
        $P_{old} = P_{new}$
        $\pi_i = \pi_i \cup X_j$
      else
        okToProceed = false
      end if
    end while
    for all X_i such that $\mathbb{B}_{i,j} = 1$, $X_j \prec X_i$, and $X_i \notin \pi_i$ **do**
      $\mathbb{B}_{i,j} = 0$
    end for
  end for

Algorithm 5 The K2GA algorithm

Require: D the data set of cases to build from, $numGenerations > 0$, $populationSize > 0$, $cullrate \in (0, 1)$, $mutateProbaility \in (0, 1)$, M the maximum indegree of a node
  $C = makeConnectionProbMatrix(D)$
  $P_0 = \emptyset$
  for $k = 1$ to $populationSize$ **do**
    $P_0 = P_0 \cup makeNewRandomMember()$
  end for
  for $gen = 1$ to $numGenerations$ **do**
    $scorePopulation(D, P_{gen}, M)$
    $P_{gen+1} = cullGeneration(P_{gen}, cullRate)$
    $P_{gen+1} = breedMembers(P_{gen+1}, populationSize)$
    $P_{gen+1} = mutateMembers(P_{gen+1}, mutateprobability, C)$
  end for

B. Heuristic Connection Probability Matrix

It is possible to use a heuristic method for setting the connection probability weights. One useful heuristic is to compute Mutual Information [9] scores between the variables and normalize the results. Using this as a preprocessing step will search more structures with highly informative connections.

Other methods to initialize this matrix could come from the work of Friedman and Koller [10] and expanded upon

by Koivisto and Sood [11], who have developed methods to compute the probability that a given feature (i.e. edge) would be present in a BBN based on a data set.

C. Adaptive Connection Probability Matrix

In each generation, many structures are considered and scored. In order to use this as feedback to help guide our search, we can use this information to help adapt our connection probability matrix for the next generation. If two edges were connected in a DAG which received a high score, the probability of them being connected in future structures should be increased, and alternatively edges in low scoring local structures should be punished.

Algorithm 6 can be added at the end of the member scoring step to perform this update.

Algorithm 6 The algorithm to adapt the connection probability matrices

Require: Initial learning rate ν_0, final learning rate ν_{MAX}, connection probability matrix for the last generation $C^{(k)}$, the maximum number of generations k_{MAX}

for $i = 1$ to N **do**
  for $j = 1$ to N **do**
    $\hat{A}_{ij} = \frac{\text{sum of member scores where } X_i \text{ and } X_j \text{ were adjacent}}{\text{number of terms in this sum}}$
  end for
end for
for $i = 1$ to N **do**
  for $j = 1$ to N **do**
    $A_{ij} = \frac{\hat{A}_{ij}}{\sum_{j=1}^{N} \hat{A}_{ij}}$
  end for
end for
$\nu^{(k)} = \nu_0 (\frac{\nu_{MAX}}{\nu 0})^{(\frac{k}{k_{MAX}})}$
for $i = 1$ to N **do**
  for $j = 1$ to N **do**
    $\hat{C}_{ij}^{(k+1)} = C_{ij}^{(k)} * (1 - \nu^{(k)}) + A_{ij} * \nu^{(k)}$
  end for
end for
for $i = 1$ to N **do**
  for $j = 1$ to N **do**
    $C_{ij}^{(k+1)} = \frac{\hat{C}_{ij}^{(k+1)}}{\sum_{j=1}^{N} \hat{C}_{ij}^{(k+1)}}$
  end for
end for

V. Numerical Results

In this section we will evaluate the performance of K2GA using four standard data sets: two generated from the well known Asia and Alarm networks and two available from the UCI machine learning repository [12].

- 10000 cases generated from the Alarm network, a 37 node network from the health care domain used for monitoring patients in intensive care which was used as a benchmark by Cooper and Herskovits [2].
- 10000 cases from the Asia Network, and 8 node network used by Lauritzen and Spiegelhalter to demonstrate their inference algorithm [13].
- The mushroom data set from the UCI machine learning repository [12]. This dataset consists of 23 variables and 8124 cases. The data set consists of 22 features of mushrooms and 1 variable determining if the mushroom is poisonous or not.
- The Microsoft anonymous web access data set from UCI contains 292 variables and 32711 cases [12]. Each variable represents a vroot on the Microsoft web site, and each case recorded the vroots that a user visited in a session on the site. We use the variant also used by Chickering [6], which consists of all of the cases and the 50 most active variables.

For each data set, we will apply K2GA ten times, five using an adaptive connection probability matrix and five without adaptation. We will show the Bayes Score (1) over time for each data set, and discuss the convergence. For the generated data sets where a ground truth is known, we will analyze the graph errors in the learned structures. We will also show the benefit of caching local scores and demonstrate the use of an adaptive connection probability matrix.

For each of the runs we will hold fixed the following parameters: 200 GA members, 50 generations, 50% of members culled each generation, 40% of members mutated per generation, 3 maximum parents per node, .05 initial learning rate, .1 final learning rate. We will also use the standard initial connection probability matrix shown in (4).

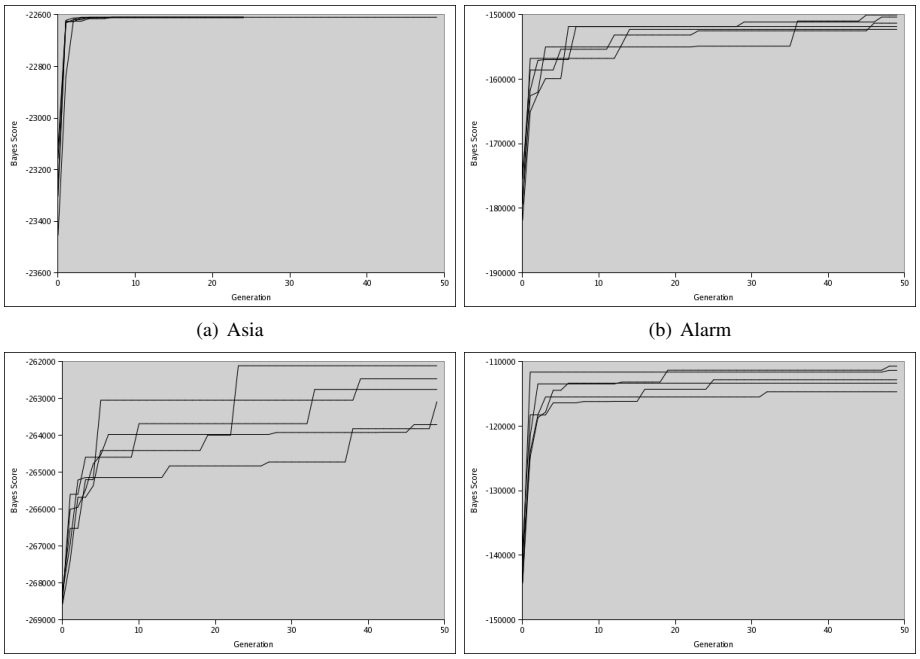
A. Convergence

Figures 2 and 3 show how the Bayes score converges over generations with and without an adaptive connection probability matrix. For a smaller network like Asia the scores converged to that of the gold standard network quickly with or without adaptation. In contrast, the worst score for the Alarm network with adaptation was better than the best score for the Alarm network without adaptation.

Table II shows the graph errors for the two networks where a ground truth network was known for each of the five adaptive runs. The score of the Asia network which had one edge error reversed was within machine precision of the score of the true network. The alarm network has 37! node orderings and an exponential number of skeletons, therefore the number of graph errors shown in Table II is very small in comparison.

B. Adaptation

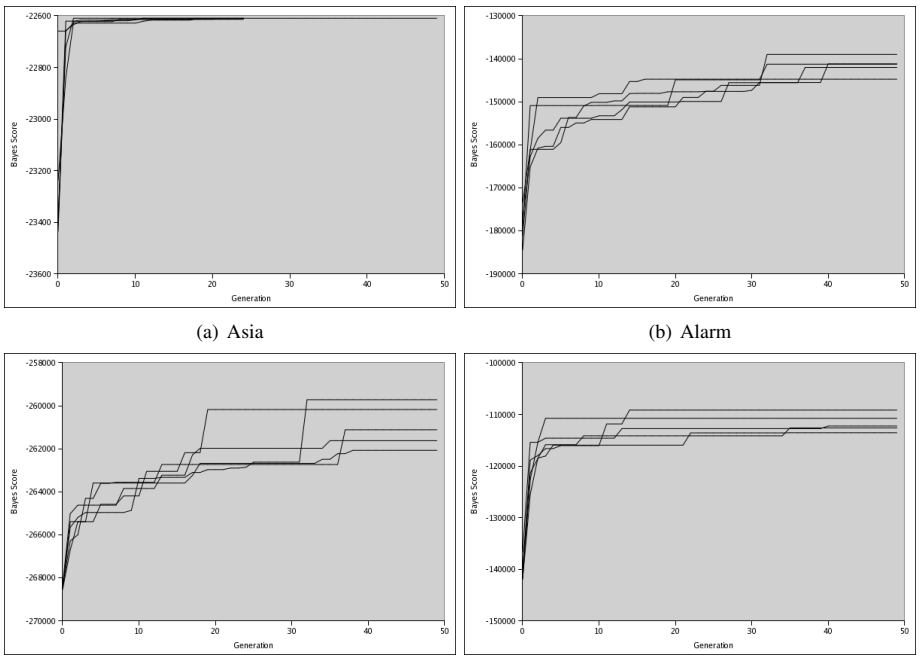
Table III shows an example of the final connection probability matrix for a run of K2GA in the Asia data set. The bold face numbers represent edges which exist in the original generating network, shown in Figure 4. This supports our claim that the nodes which are truly connected in the network have larger probabilities of being connected during the search, while ensuring that any two nodes have a reasonable probability of being connected. Using adaptation no potential



(a) Asia (b) Alarm

(c) Microsoft (d) Mushroom

Fig. 2. The Bayes Scores over time for 5 runs of the example data sets without an adaptive connection probability matrix



(a) Asia (b) Alarm

(c) Microsoft (d) Mushroom

Fig. 3. The Bayes Scores over time for 5 runs of the example data sets with an adaptive connection probability matrix

	Visit Asia	Tuberculosis	Smoking	Cancer	TbOrCa	XRay	Bronchitis	Dyspnea
Visit Asia		**.871**	.0103	.0818	.00164	.0155	.0114	.00816
Tuberculosis	**.424**		.0581	.167	**.113**	.154	.0108	.0737
Smoking	.0051	.0596		**.320**	.0512	.0699	**.3**	.195
Cancer	.0397	.175	**.312**		**.166**	.13	.11	.0671
TbOrCa	.00068	**.107**	.0422	**.149**		**.366**	.14	**.195**
XRay	.00775	.173	.0685	.138	**.443**		.0704	.0989
Bronchitis	.00556	.0114	**.293**	.111	.171	.0654		**.343**
Dyspnea	.00394	.0776	.188	.0662	**.234**	.0937	**.336**	

TABLE III
THE FINAL CONNECTION PROBABILITY MATRIX FOR AN ADAPTIVE SOLVE OF THE ASIA DATA SET

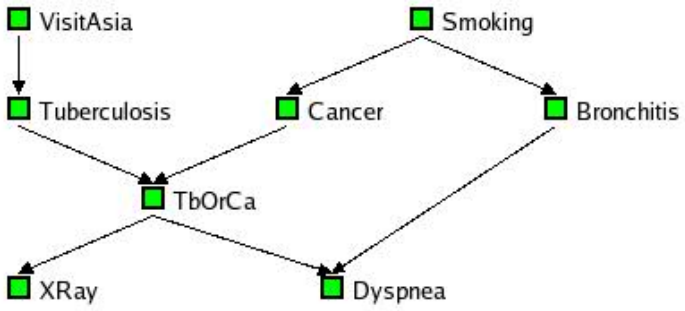


Fig. 4. The true structure of the Asia network

Asia			
trial	# reversed	# missing	# extra
1	0	0	0
2	1	0	0
3	0	0	0
4	1	0	0
5	1	0	0

Alarm			
trial	# reversed	# missing	# extra
1	7	31	19
2	4	33	29
3	7	32	26
4	5	31	24
5	6	33	29

TABLE II
EDGE GRAPH ERRORS FOR ASIA AND ALARM ON EACH RUN

Alarm		
trial	# local scores	# from cache
1	1361356	1325017
2	1374649	1337228
3	1379371	1341223
4	1382948	1346861
5	1367617	1330671

Mushroom		
trial	# local scores	# from cache
1	1063521	1041366
2	1066981	1045066
3	1066239	1043945
4	1072192	1048232
5	1070833	1046588

TABLE IV
COMPARING THE NUMBER OF LOCAL SCORES VERSUS THE NUMBER PULLED FROM THE SCORE CACHE

graph structures are eliminated from the search, but the search is guided towards the best structures.

C. Caching

Table IV shows the benefit of caching local scores. While caching has no effect on the search algorithm, it does have a significant effect on running time. While scoring the Alarm network, 97.3% of the local scores were pulled from the cache. While scoring the Mushroom network, 97.8% of the local scores were pulled from the cache. This represents a huge speed increase over recomputing those local scores, at the cost of a relatively small amount of memory.

VI. COMPARISON TO RELATED METHODS

We are now able to summarize the key differences between K2GA and other common model selection techniques.

Although genetic algorithms for model selection have been used in the past, K2GA presents several advantages over previous methods. The work of Hsu [7] searches topological orderings of variable and applies a full K2 algorithm at each ordering. A full K2 search is much less efficient than the modified K2 algorithm. The genetic algorithm of Larrañaga [8] searches both orderings and skeletons, but does so in

a way that can allow illegal structures requiring a repair operator before scoring. K2GA's member representation, use of a modified K2 algorithm, and score guided breeding and mutation represents a significant improvement over these previous genetic algorithms.

Hill climbing searches are a common technique for learning the structure of Bayesian networks [14]. A hill climbing search begins with a Bayesian network structure $\mathbb{G}_k$, and performs a series of edge additions, deletions, and typically some edge reversals to find the neighbors of $\mathbb{G}_k$, then applies the operation which resulted in the highest scoring neighbor to produce $\mathbb{G}_{k+1}$. This is continued until no neighbors produce a higher score. These methods are very simple but often get trapped in local optima.

Christopher Meek's 1997 thesis [15] addressed some of the problems with greedy search. Meek presented a 2 stage search algorithm, the first consisting of a sequence of edge additions and removals, and the second consisting of a series of edge deletions, each of which were chosen in such a way to avoid local optima. K2GA can be seen as a parallel to this idea where *each generation* performs a series of additions and reversals, then deletions. Each crossover and mutation adds and reverses edges. Each time a structure is scored the modified K2 algorithm removes edges.

Max Chickering's extension of the work of Meek [6], [16] provides a major improvement to Meek's greedy search algorithm. Chickering presents GES, an algorithm which searches the space of Markov equivalence classes of Bayesian networks rather than the space of all DAG's.

The work of Meek and Chickering represents a significant contribution to model selection. Future extensions of K2GA will extend many of the concepts developed by Meek and Chickering in our genetic algorithms based framework.

VII. Conclusions

The algorithm which we have presented makes several new contributions to automated model selection.

- K2GA searches the topological ordering and adjacency matrix simultaneously
- Since K2GA members cannot contain cycles, no repair operator is needed
- The breeding function is guided by structure score
- The self optimized members are used in future generations
- A heuristic connection probability matrix can be used to guide the search
- An adaptive connection probability matrix can be used to decrease time to convergence

These new concepts make K2GA both a viable model selection algorithm and an important extension to current research.

References

[1] P. Spirtes, C. Glymour, and R. Scheines, *Causation, Prediction, and Search, Second Edition (Adaptive Computation and Machine Learning)*. The MIT Press, 2001. [Online]. Available: http://www.amazon.fr/exec/obidos/ASIN/0262194406/citeulike04-21

[2] E. Herskovits and G. Cooper, "Kutató: An entropy-driven system for construction of probabilistic expert systems from databases," in *Proceedings of the 6th Annual Conference on Uncertainty in Artificial Intelligence (UAI-91)*. New York, NY: Elsevier Science Publishing Company, Inc., 1991.

[3] D. Heckerman, "A tutorial on learning with bayesian networks," Microsoft Research, Redmond, Washington, Tech. Rep., 1995, revised June 96. [Online]. Available: citeseer.ist.psu.edu/article/heckerman96tutorial.html

[4] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pretince Hall, 1995.

[5] J. Pearl, *Causality*. Cambridge Universtiy Press, 2000.

[6] D. M. Chickering, "Learning equivalence classes of bayesian-network structures." *Journal of Machine Learning Research*, vol. 2, pp. 445–498, 2002.

[7] 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 *GECCO '02: Proceedings of the Genetic and Evolutionary Computation Conference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 383–390.

[8] P. Larrañaga and M. Poza, "Structure learning of bayesian networks by genetic algorithms: A performance analysis of control parameters," *IEEE Journal on Pattern Analysis and Machine Intelligence*, vol. 18, no. 9, pp. 912–926, 1996. [Online]. Available: citeseer.ist.psu.edu/larranaga94structure.html

[9] C. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, 1948.

[10] N. Friedman and D. Koller, "Being bayesian about network structure," in *Proceedings of the 16th Annual Conference on Uncertainty in Artificial Intelligence (UAI-00)*. San Francisco, CA: Morgan Kaufmann, 2000, pp. 201–2.

[11] M. Koivisto and K. Sood, "Exact Bayesian structure discovery in Bayesian networks," *Journal of Machine Learning Research*, vol. 5, pp. 549–573, 2004. [Online]. Available: citeseer.ist.psu.edu/koivisto04exact.html

[12] C. B. D.J. Newman, S. Hettich and C. Merz, "UCI repository of machine learning databases," 1998. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[13] S. Lauritzen and D. Spiegelhalter, "Local computation with probabilities in graphical structures and their applications to expert systems," *Journal of the Royal Statistical Society*, 1988.

[14] R. E. Neapolitan, *Learning Bayesian Networks*. Prentice Hall, 2004.

[15] C. Meek, "Graphical models: selecting causal and statistical models," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, 1997.

[16] D. M. Chickering, "Optimal structure identification with greedy search," *Journal of Machine Learning Research*, vol. 3, pp. 507–554, 2002.