# Improving High-Dimensional Bayesian Network Structure Learning by Exploiting Search Space Information

Technical Report 06-49

Avi Herscovici Oliver Brock
Department of Computer Science
University of Massachusetts Amherst

October 2, 2006

**Abstract**

Bayesian networks are frequently used to model statistical dependencies in data. Without prior knowledge of dependencies in the data, the structure of a Bayesian network is learned from the data. Bayesian network structure learning is commonly posed as an optimization problem where search is used to find structures that maximize a scoring function. Since the structure search space is super-exponential in the number of variables in a network, heuristics are applied to constrain the search space of high-dimensional networks. Greedy hill climbing is then applied in the reduced search space. The constrained search space of high-dimensional networks contains many local maxima that greedy hill climbing cannot overcome. This issue has only been addressed by augmenting greedy search with TABU lists or random moves. This is not a holistic solution to the problem.

By using a search algorithm that is global in nature, we are not confined to results in a particular region of the search space, like previous approaches. We present Model-Based Search (MBS) [1] applied to Bayesian network structure learning. MBS uses information gained during search to explore promising search space regions. Maintaining this search space information keeps a global view of the search task and helps find structures at higher maxima than greedy hill climbing. We show that MBS performs better than hill climbing in the Max-Min Parents and Children (MMPC) [30] search space and can find better high-dimensional network structures than other leading structure learning algorithms.

## 1 Introduction

The scientific community is quickly gathering a plethora of new data sets that can be analyzed to form significant scientific conclusions. For example, DNA microarray data is collected for analysis since it contains information on the interactions of proteins. Economic data is analyzed to form models that can be used to forecast events such as economic recessions. Initially, understanding relationships in these domains was done through experimentation, which is usually costly and inefficient. When computers became commonplace, scientists began using computational methods that could exploit the newly available computational power to extract meaningful information from data sets.

A data set is a collection of samples of the state of different random variables at a particular time. By statistically analyzing these data sets we can estimate the distribution of the variables sampled in the data and predict the state of a variable with knowledge of the variables on which it is dependent. Without prior knowledge of dependencies between the variables, it is a computationally expensive task to learn the dependencies and estimate the distributions. The complexity and difficulty in learning dependencies in data is due to the following two factors: how many data samples are available and the amount of random variables the data set describes. This thesis focuses on the amount of random variables the data set describes, formally known as the dimensionality of the data set. A high-dimensional data set is composed of samples from many random variables. High-dimensional data sets are common in biology, social networks, and other domains where a large amount of entities are interacting.

A common method for learning dependencies between random variables is to train a Bayesian network on data sampled from the distribution of these variables. A Bayesian network [25] is a graphical model that compactly expresses the joint probability distribution of a set of random variables. More importantly, it allows the user to visually infer dependencies between the random variables from the model. An example in the biology domain is a Bayesian network, trained from genomic data, that expresses dependencies between proteins [16]. The graphical model of a Bayesian network is a directed acyclic graph that contains a node for each random variable. Dependencies among the variables in the network are expressed by edges. The set of edges in a Bayesian network are referred to as the network structure. Without prior knowledge of the network structure (the graphical representation of dependencies), the structure that best fits the data must be learned from the data.

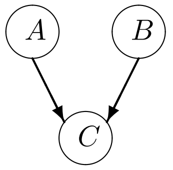
To learn a network structure computationally, we first formulate a search space of possible structures, which consists of all the different edge combinations that form an acyclic graph. This search space is vast even for a small amount of variables since an increase in the amount of nodes leads to a super exponential [27] increase in the amount of possible structures. Finding the optimal Bayesian network in the search space of all possible structures was shown to be NP-*hard* [5]. Searching the network structure search space by exhaustively evaluating network structures is infeasible. The best network structure must be approximated.

Approximating the best network structure is commonly posed as an optimization task that is solved by a search technique and a scoring function which evaluates network structures. While a particular scoring function has not been singled out as better than the others, a multitude of search algorithms have been used in search-based methods with varying success. Greedy search algorithms were found to perform well on low-dimensional data sets. They work by performing operations on an empty network that cause the greatest score improvement at each iteration of search. In small networks, greedy search returns near optimal results because the search space is not very complex and contains few local maxima. The performance of greedy search on high-dimensional data sets begins to degrade because the search space is more complex.

To learn high-dimensional networks, researchers have used constraint-based techniques to reduce the structure search space. Constraint-based techniques perform a statistical test on the network variables and create a reduced search space which does not contain node pairs that are deemed independent by the statistical test. Greedy search is then applied in the reduced search space. Using greedy search hampers search quality because of local maxima in the constrained search space that greedy search cannot overcome. Greedy search only explores a *local* region of a complex search space, which will only give locally optimal results. Search is confined to these local regions leaving more promising regions unexplored for higher maxima. By using a global approach to search, we can encounter higher quality regions of the search space that greedy search leaves unexplored.

Model-Based Search (MBS), introduced in [1], has been successful in optimizing energy functions in protein structure prediction. MBS builds a compact model that approximates the landscape of a search space. MBS uses this model to guide search towards promising regions of the search space. This thesis is a culmination of the research performed to improve high-dimensional Bayesian network structure learning using MBS. We present a search-based structure learning algorithm which uses MBS to find high scoring network structures in the Max-Min Parents and Children (MMPC) search space [30].

Background on Bayesian networks and in particular learning Bayesian networks from data follows the introduction. Related works that influenced our research are mentioned as an inspiration to our approach and



| $A$ | $B$ | $C$ |
|---|---|---|
| 0 | 0 | 1 |
| 2 | 1 | 1 |
| 1 | 1 | 0 |
| 2 | 0 | 2 |
| ... | ... | ... |

Figure 1: A Bayesian network of 3 discrete variables. Each variable has a domain size of 3 meaning each variable can take on three discrete states. An example data table is also provided.

as a reference to the reader. The Model-Based Search algorithm is analyzed, and its implementation specific to Bayesian networks is described. Experiments varying the MBS parameters are presented to show the flexibility of MBS. Also the benefits of MBS over unoptimized distributed search is shown empirically. Last, we show using two different quality metrics that MBS returns higher quality high-dimensional Bayesian network structures than other leading structure learning algorithms. MBS clearly outperforms greedy search in the MMPC search space. The results obtained are promising for finding accurate relationships in data sets, yet there is still room for further progress in structure learning. We end by suggesting improvements to MBS that can be made in future work.

## 2 Bayesian Networks

A Bayesian network, $B$, is composed of a pair, $(G, \theta)$, where $G$ is the network structure and $\theta$ is a set of parameters. A Bayesian network's structure $G = (V, E)$ is a directed acyclic graph (DAG) which describes conditional independence properties between random variables in the set $V$. The edges in the network, $E$, correspond to conditional dependencies through the following property: every random variable is independent of its non-descendants conditioned on its parents [25]. We call a variable $V_1$ the parent of variable $V_2$ if there is an edge from $V_1$ to $V_2$. The set of parameters, $\theta$, represent the probabilities of each random variable given its parent node set:

$$\theta_i = P(V_i | V_{Parents(i)})$$

The joint probability of the variables in a network is defined as follows:

$$P(V_1, \ldots, V_n) = \prod_{i=1}^{n} P(V_i | V_{Parents(i)})$$

There are many advantages to using Bayesian networks to express joint probabilities of random variables and dependencies between random variables. By explicitly stating and utilizing the dependency information in the model, one can save computation of insignificant probabilities between independent variables. The dependency information in the network is also useful for decision theory and allows for expert information to be included. The parameters of a Bayesian network are always semantically meaningful as opposed to black box models like neural networks. The output of a Bayesian network is consistent, meaning it will always return the same answer if the model has the same parameters. Last, a Bayesian network can contain continuous variables, discrete variables, or a mix which makes the model flexible for different applications.

An example Bayesian network along with a data set sampled from the network's variables is shown in Figure 1. The variables are discrete and can be in three states (0-2). Continuous random variables can be trained and queried like discrete random variables, but they are beyond the scope of this thesis. A common way to reduce the complexity added by continuous variables is to discretize the continuous data and proceed using discrete variables.

In the example network, the probability distribution of random variable C is dependent on the values of A and B. Since this example is discrete, the probability distribution of C is expressed as a conditional probability table listing the probabilities of each value of C (0 through 2) given each combination of values for variables A and B. The table for C would then have 27 entries if each probability is recorded or 18 if only the essential probabilities were the recorded, where the rest can be easily calculated since the sum of the probabilities of all the settings of variable C per combination of A and B is equal to 1. For example: $P(C = 0|A = 0, B = 1) + P(C = 1|A = 0, B = 1) + P(C = 2|A = 0, B = 1) = 1$. The tables of A and B would only have 3 entries each in the complete table since they are not dependent on any other variable and can assume 3 different values. With complete probability tables, we can query a Bayesian network for the probabilities of particular states of the network variables. We can also learn the probability tables and the network structure from data.

## 3 Learning Bayesian Networks

The learning problem for Bayesian networks has two components. A network with a defined structure $G$ but without specified parameters requires computing the parameters from a data set with samples of the network variables. The parameter learning problem can be formally stated as maximizing the likelihood of the parameters $\theta$ given observed data $D$ and a network structure $G$. Given only data and no network structure, we must learn the structure $G$ of the network and then learn the parameters. Our variable set $V$ is defined by the random variables that are sampled in the data; hence, structure learning involves connecting the variables in $G$ (i.e constructing the set of edges $E$). More formally, the structure learning problem is to maximize the likelihood of the network structure $G$ given observed data $D$. Learning with incomplete data is possible and is a heavily researched topic, but this thesis only considers complete data sets and refers the reader to other literature about the topic [10, 26, 7].

Estimating the parameters of the network (parameter learning) means to estimate the local distribution function of each random variable in a Bayesian network. Each local distribution is essentially a probabilistic classification (in the discrete case) or a regression (in the continuous case) function. Using a maximum likelihood approach [14] to estimate the parameters of discrete variables is straightforward since a parameter can be computed by counting occurrences of a state in the data set, which is how many times the variable X is of a particular state, and normalizing by the data sample size, which is the number of rows in the data set or the amount of total samples per variable. A Bayesian approach can also be used to estimate parameters by using a prior, most commonly a Dirichlet prior for multinomial variables. Then the Maximum a Posteriori (MAP) estimate, which in the multinomial case is also Dirichlet, must be computed using the prior.

Structure learning has to be approached differently since computing the exact posterior over Bayesian network structures is intractable for large networks. Search and score approaches are the most common methods in finding network structures that fit a data set. They pose structure learning the structure as an optimization problem. The search component is an algorithm with the goal of identifying high scoring Bayesian network structures. The scoring function returns a score indicating how well a structure fits the given data. When considering the approach holistically, the scoring function creates a landscape over the different network structures that is traversed by the search method. The search method is used to find the the global maximum of this landscape, which indicates the structure that best fits the given data.

## 3.1 Scoring Functions

Common scoring functions used to evaluate Bayesian network structures are the MDL, BIC, and the BDe(u) score. The Minimum Description Length score, MDL, is computed from a description length formula, which is minimized using a search procedure. Structure learning using the MDL score was presented in [19]. The Bayesian Information Criterion (BIC) score is also commonly used and is exactly minus the MDL score [14]. The BDe score is derived in [14]. The BDeu score is an extension of the BDe metric where the prior joint space, conditioned on the network, is uniform. The BDeu score is quite common in other literature and there have not been any definitive studies showing the merits of one scoring function over another. We use the BDeu score because it has the following attractive properties. With complete data the BDeu score is decomposable and can be expressed as follows:

$$Score(G, D) = \sum_i Score(V_i, Parents(V_i))$$

The implication of decomposability in the BDeu score is that the score only needs to be locally recomputed when a change to the network occurs. For example, if a 3 variable network with the numbers 1-3 as variable names were changed by adding an edge from node 1 to 3, then only node 3's score component needs to be recomputed (since its parent set changed). Nodes 1 and 2 maintain the same contribution to the score. For large networks this saves an immense amount of computation. Another important property of the BDeu score is score equivalence. When two graphs are members of the same *Markov equivalency class*, they both represent the same joint probability distribution. This implies that we only need to find a structure in the same equivalency class as the optimal graph and the structure will be statistically equivalent to the optimal graph.

## 3.2 Search

Search techniques used in structure learning are predominantly greedy, meaning they require heuristic information about subsequent states when choosing their next state. This is a computationally expensive requirement. Stochastic search, which uses randomness in selecting a subsequent state, is less computationally expensive but does not return consistent results. The stochastic search Metropolis-Hastings Monte Carlo [22] is used [21] as a search method in search-based structure learning. The most common search method [3] applied in search and score is greedy hill climbing, which performs local operations on the network that lead to the best change in score until no more positive changes can be made. Hill climbing can be augmented with a TABU list [12] or random restarts to try and escape local maxima. Due to its greediness, hill climbing is computationally expensive and cannot be run on high-dimensional networks.

Researchers noted that searching DAG-space is slightly redundant, since some local moves result in a graph that is in the same equivalence class as its predecessor in search. Greedy Equivalence Search (GES) [6] performs greedy search in the the space of equivalence classes and represents an equivalence class by a partially directed acyclic graph (PDAG). GES theoretically finds the most probable network in the sample limit if the distribution of the data is faithful. In practice, GES is only locally optimal, and there were attempts [24] to overcome local maxima during greedy search by introducing randomness into GES. This approach improves results but does not solve the problem of local maxima holistically. There are also approaches that search over orderings of network variables instead of graphs. In a recent paper [8], the authors use Markov Chain Monte Carlo (MCMC) to search over orderings and compute a posterior over features (i.e the posterior probability over models that contain a particular feature). There was also work done on searching orderings with genetic algorithms [20].

The class of algorithms most successful in learning high-dimensional networks use a hybrid technique of reducing the search space and then performing search. Sparse Candidate [9] was the first successful algorithm that was applied to a network with hundreds of nodes. Sparse Candidate constrains the search to find up to $k$ candidate parents for each variable by using heuristics. A subnetwork is then created using hill climbing and the entire process repeats until the candidate sets do not change or there is no improvement in the network score. Max-Min Hill Climbing (MMHC) [31] improves upon Sparse Candidate by offering a sound method in finding parents and children of a variable and not limiting the number of parents per variable. The method is sound because it performs sound conditional independence tests, which means it returns only statistically equivalent networks that are consistent with the independence test. By using the Max-Min heuristic only a single iteration is required for each variable to find candidate parents and children. The component which creates the parents and children search space is called Max-Min Parents and Children (MMPC) [30]. A greedy search is then performed in the MMPC search space to orient the edges in the search space. Due to the previously mentioned benefits of the MMPC search space over Sparse Candidate's, we perform search in the MMPC search space, which is described in the following section.

## 3.3 MMPC

MMPC returns a candidate set $PC$ of parents and children for a variable in a Bayesian network given data $D$. Running MMPC on each variable of a network produces a constrained search space or "an unoriented skeleton" [31] of the Bayesian network. The true MMPC algorithm improves on the original MMPC algorithm called $\overline{MMPC}$. $\overline{MMPC}$ works in two phases. First we define the minimum association between two variables given a set of variables. The minimum association for variables $X$ and $T$ given a set $Z$ and a dependency metric $dep$ (where $dep$ is the inverse of an independence metric) is defined as follows:

$$MinAssoc(X, T, Z, dep) = \min_{S \subseteq Z} dep(X, T|S)$$

MinAssoc returns the lowest score for all subsets $S$ of $Z$. The first phase of $\overline{MMPC}$ creates the set $CPC$ for our target variable $T$ and augments it according to the Max-Min heuristic. The Max-Min heuristic favors variables that maximize the minimum association with $T$ with respect to $CPC$. The intuition behind the Max-Min heuristic is to find the variables that are dependent on $T$ no matter on which subset of $CPC$ we condition. The second phase of $\overline{MMPC}$ attempts to remove false positives by testing for independence on all subsets of $CPC$. If this holds for some subset then the tested variable is removed from $CPC$. The MMPC algorithm runs $\overline{MMPC}$ on each variable and removes more false positives by checking the following: for every member $X$ from a target variable $T$'s candidate set $CPC$, we remove $X$ from $CPC$ if $T$ is not a member of $X$'s CPC set. The correctness of the algorithm is shown in [31]. The implementation of MMPC uses the $\chi^2$ test for conditional independence.

## 3.4 Other Approaches to Structure Learning

A different set of approaches to structure learning is known as constraint-based. A constraint based method uses statistical dependence measures, for example the $\chi^2$ test, to discover dependencies between variables in the network. Dependency information is then used to create constraints on the network structure, since variables found to be independent cannot have an edge between them. The SGS algorithm [28] and PC algorithm [29] are examples of constraint-based algorithms. A more recently presented constraint-based algorithm [2], Three Phase Dependency Analysis, uses techniques from information theory to test for dependencies and perform search.

Structure learning algorithms also use constrained search to reduce computation time. Optimal Reinsertion [23] is a constrained search that uses special data structures to be more efficient. It starts from a particular network and then node by node removes edges and reinserts higher scoring edges by searching through parent and children sets. An exact method for finding a network structure is proposed in [18], but it is only feasible on networks that have under a hundred variables with restrictions on the structures. Variational methods for structure learning were proposed in [17]. There is also an approach [13] that learns networks using frequent sets.

## 4 Model-Based Search

As discussed in the previous section, a common approach to learning high-dimensional networks using search and score methods is to reduce the search space in order to use low dimensional search techniques efficiently. Greedy search is quite effective for low dimensional problems, and the most intuitive solution is to reduce our problem to something we already know how to solve effectively. Unfortunately, the main weakness of greedy search, its inability to escape local maxima, becomes a larger issue in the reduced search space. Solutions like adding TABU lists or using random restarts do not address the problem directly. We are still performing local search with these methods, and the results cannot substantially improve until we can efficiently explore different regions of the search space. Model-Based Search (MBS) is designed with a global perspective in mind. By using information we gain during search and some moderate assumptions about the search space, we can direct search towards promising regions of the search space where we encounter higher maxima.

In most cases, a high-dimensional search space will contain a few promising areas fairly near each other and many low scoring regions that are far from the optimal solution. By focusing on promising regions in the search space, we avoid running through many similar and low scoring structures. MBS constructs a model of the search space and *discards* low scoring areas. This is contrary to a TABU search where low scoring structures are maintained. MBS maintains a model of regions in the search space that it determines promising. It then uses the knowledge of promising regions to further explore the search space and improve the model until search does not discover higher scoring regions.

Updating the model of the search space is done by sampling. MBS heavily samples regions that receive high scores from a heuristic and discards samples that are low scoring. This adaptive sampling can also be viewed as a communication between samples, where one sample sends a message to a few others to help explore a promising region. This approach helps avoid local maxima. Figure 2 gives an illustration of a sample in a 5-sample model that encounters a maximum and is later moved to aid search when a more promising region is found.

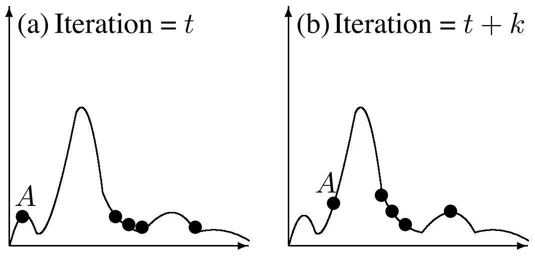


Figure 2: (a) At iteration $t$, sample $A$ is stuck in a local maximum. It will either remain there if the other samples have not found a better region to search and claim it is the global maximum or (b) join the other samples at iteration $t + k$ when they encounter a more promising area.

Using Model-Based Search requires the following: a sampling method for exploration, an initial sampling strategy, an evaluation function, and a distance metric that forms the landscape of the evaluation function. Any sampling method can be plugged in to explore the search space. It is also helpful to also pick a method for sample improvement if the sampling does not include a score improvement criteria. Metropolis Hastings Monte Carlo [22], which rejects samples that score below a threshold, is commonly used. The initial sampling strategy should construct the initial model using domain information, heuristics, or by sampling uniformly.

The distance metric has to create a search landscape that meets the following assumption for MBS to work effectively: the search space has to be continuous, in the sense that high scoring samples will be near other "similar" high scoring samples. The search space cannot have one extremely good sample that is far away from all other samples because our model assumes that sampling in high scoring regions will increase the likelihood of encountering high scoring structures. Figure 3 illustrates a good and bad landscape for MBS.

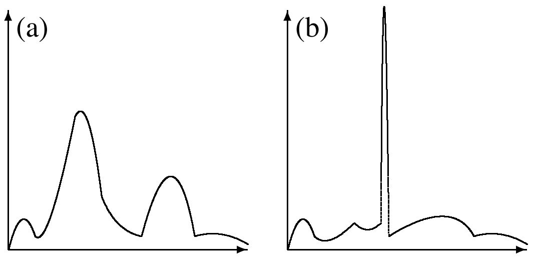


Figure 3: (a) A good landscape for Model-Based Search, where the landscape is continuous and contains several wide hills where sampling in a region will lead us to the maximum. (b) A bad example where there is one very narrow maximum far from any other high scoring regions, so sampling in the region will not lead us to the global maximum.

The evaluation function takes two metrics into consideration. First is the quality of the sample given by some energy or scoring function, which we call the sample score. Second is a metric used to help maintain distant samples, which prevents search from becoming too concentrated. This metric is called the radius and is defined as follows: the shortest distance to a sample with a higher score, normalized by the largest distance. A weighted sum of these metrics is known as the model score and is used to evaluate the fitness of the samples in our model.

Model-Based Search Algorithm

1. Create model by allocating $n$ initial samples
2. Compute model score of each sample
3. Prune low scoring samples
4. Allocate new samples to sampling method
5. Sample around high scoring samples
6. Test for convergence and either go to 2 or terminate

Figure 4: The steps of the Model-Based Search algorithm

Using our sampling methods, evaluation function, and distance metric as building blocks, we can implement MBS using the steps in Figure 4. The next section describes the implementation of these steps applied to Bayesian network structure learning.

## 4.1 Model-Based Search for Bayesian Network Structure Learning

We conduct Model-Based Search in the Max-Min Parents and Children (MMPC) [30] search space, which is a reduced space of candidate parents and children for each variable (described in Section 3.3). Using a reduced search space allows for hill climbing to be used in sample improvement, which results in a less strict, efficient initial sampling.

### 4.1.1 Initial Model

Since the search space we are modeling is large, we need to spread our initial samples. If the samples are too concentrated, the search may ignore a region that it cannot reach in future iterations. In using MBS to search an energy landscape of protein configurations [1], the initial sampling of the search space is done uniformly. There has been work done [15] on uniformly generating sparse DAGs. An efficient implementation of uniform DAG generation would benefit MBS except when using hill climbing for sample improvement. Using hill climbing in sample improvement already guides the samples towards the most promising set of regions in the search space. This means spending time sampling uniformly would be wasteful because we know a priori that the majority of high scoring structures will have a significant amount of common edges. Our implementation of initial random graphs simply maintains randomness in the initial samples that will eventually contain the high scoring features encountered by Monte Carlo or hill climbing. Random DAG generation is implemented as follows:

- Create random ordering of $|V|$ variables
- Connect each member of the list up to a maximum of $k$ parents

By creating a strict ordering and connecting each variable to a variable further in the list, the graph is guaranteed to be acyclic. There are also two sources of randomness: a random ordering and random connections adhering to the ordering. Connecting each node to a maximum of $k$ (3 in our implementation) parents maintains sparseness. Initial sparseness is beneficial since heavily connected graphs will usually contain spurious edges which lead search to a local maximum.

### 4.1.2 Distance Metric

Since we only want to explore regions of the search space that could lead us to the global maximum, we maintain samples in high scoring regions while also keeping samples that are far away from the current samples to prevent our search from being too concentrated. To perform this type of sampling we need to define a distance metric between two samples. The distance metric is a critical component as it determines the landscape of our search space. We also want to maintain the property that nearby samples have similar features.

A simple distance metric implementation is the sum of the exclusive-or of two networks' adjacency matrices. This is essentially accumulating the difference in the DAG structure. This basic distance metric is computed easily but contains a few flaws. Reversed edges are counted twice in making two samples dissimilar whereas in some networks (e.g ones in the same equivalence class) the orientation of an edge bares no effect on the statistical meaning of the model. An easy fix is to discount reversed edges. A more computationally expensive solution is to convert each structure to a partially directed acyclic graph (PDAG), which represents a structure's respective equivalence class, and compute the difference between the two

PDAGs. We can also go a step further and include statistical similarities between two Bayesian networks. This can be done using a statistical test, such as mutual information. Mutual information is defined as:

$$MI(A,B) = \sum_{a,b} P(a,b) log \frac{P(a,b)}{P(a)P(b)}$$

P(a) is computed by normalizing the count of "a" in the data set. By factoring in a missing edge's statistical effect on the model, two samples are different if the set of edges in their intersection contributes to the statistical meaning of their respective models. A missing edge with high mutual information will have a large contribution towards the sample (BDeu) score and is therefore more significant when comparing a structure that contains the edge and a structure that does not.

### 4.1.3 Allocation of New Samples

To explore around our promising samples, each region in the model is assigned a particular number of new samples to be generated by the sampling methods. The amount of samples per region is determined by the model score (described in the previous section) of the sample representing the respective region. The sample score component of the model score in our implementation is the BDeu score, and the distance metric component is the difference of two DAGs weighted by mutual information. Regions where the model score is the highest are sampled the heaviest. The total number of samples is kept consistent with the initial number of samples by pruning (section 4.1.6).

### 4.1.4 Sampling Regions

Once the number of samples has been allocated to a region, the search space is sampled in this region. This is done by running Metropolis Monte Carlo [22] from each sample. Metropolis Monte Carlo explores the region by performing a random walk using add edge, delete edge, or reverse edge as operations. It accepts higher scoring steps and some lower scoring ones based on a Boltzmann function:

$$P(t) = e^{\frac{score_{new} - score_{old}}{KT}}$$

By changing the parameter $T$, one can adjust the threshold for accepting a low scoring sample.

### 4.1.5 Sample Improvement

After sampling there is a sample improvement step. Monte Carlo improves scores while performing its random walk, but even a few steps of hill climbing can add high scoring edges that Monte Carlo may take a long time to encounter randomly. This step is specific to MBS in Bayesian Network structure learning and is motivated by the property that high scoring structures will usually contain a significant amount of common edges. The sample improvement algorithm is greedy hill climbing (described in section 3) with the following technique to save computation. Since a local move only affects a local component of the scoring function as discussed in the background section, we only need to keep track of the best local move per node. Initially this requires computing the score increase for every possible edge in the search space, but after this initial computation we only have to recompute the best move for the node that was affected by the most recent move. We then check a data structure that caches the maximum score improvement from every local operation and perform the best operation.

### 4.1.6 Model Pruning

Since we want to keep our model at a reasonable size, there is some discarding of low scoring samples. We discard samples that correspond to regions of the search space that are not promising for the search to explore further. The remaining samples after pruning indicate the regions that will be explored in this iteration of MBS.

### 4.1.7 Convergence

We define convergence as stagnation in the improvement of the model. MBS is set to halt after $k$ number of iterations where the maximum score does not improve. There may be more effective measures of convergence, but to keep our experiments consistent, we follow the measure of convergence for the algorithms we test against.

## 4.2 Implementation Detail

### 4.2.1 Checking for Cycles

Since Bayesian networks adhere to the DAG constraint then we must check for cycles after making any changes to a graph. Checking for cycles must be done efficiently since it must be done so often. In our implementation, we store extra information in order to perform cycle checking in less time. We follow the algorithm of [11], where maintaining an ancestor matrix that is the size of the adjacency matrix of the graph is sufficient for a significant time boot. Checking for cycles after edge addition and deletion is constant, and checking for a cycle after an edge reversal is linear. Updating the ancestor matrix after an accepted edge addition is linear, while removal and reversal are quadratic.

### 4.2.2 Causal Explorer

Causal Explorer is a Matlab Bayesian network structure learning package developed at the University of Vanderbilt's Discovery Systems Laboratory and contains many of the implementations in [31]. The MBS implementation was done using functions made available in Causal Explorer, so the other algorithms implemented could be closely compared with MBS.

## 4.3 Variations on MBS

The implementation of MBS discussed in this paper performs search in the Max-Min Parents and Children search space and creates initial samples in this search space as well. Using a reduced search space for this particular implementation of MBS is essential because we use hill climbing as a sample improvement method. The advantage of this method is if the reduced search space contains most of the edges in the optimal graph, we can improve the scores of our samples more efficiently and still encounter a near-optimal solution. The process is more efficient because we are not waiting for Monte Carlo to randomly encounter the edges that most improve the score. Using only Monte Carlo also leads to samples with many edges that do not significantly improve the score and are not in the optimal graph. These are also dense graphs near maxima in the search space.

Since the heuristics used for constraining the structure search space are not perfect, we can only find the best structure in our reduced search space. This may not be the optimal structure when searching the original search space. A variation of the current implementation of MBS has the potential for better results but cannot rely on greediness. In the variation, the algorithm creates initial samples in the MMPC search space but then performs the search component in the entire DAG search space. For this approach to be tractable, sample improvement by greedy search must be removed from MBS. Maintaining a greedy search space for each sample in the original unconstrained search space is too expensive, so sample improvement must be less greedy (e.g relying on Monte Carlo). We eventually encounter the issues discussed above with Monte Carlo, so a high performing implementation remains a challenge.

## 5 Experiments

In the first experimental section, we present experiments varying the parameters of Model-Based Search, compare it to a distributed search that does not share information, and show it is a low variance stochastic search. In the second experimental section, we compare the performance of Model-Based Search with the Hill Climbing (augmented with a TABU list) and the Max-Min Hill Climbing algorithms implemented in Causal Explorer. For the Gene network, hill climbing is replaced with the Sparse Candidate algorithm also implemented in Causal Explorer. MBS is shown to consistently find higher quality networks on high-dimensional networks using two separate metrics. The data used in all experiments is generated from a known network, so the learned network can be compared with the optimal result. Table 1 shows the networks from which the synthetic data sets were generated and their properties.

| DATA SET | VARIABLES | EDGES | DOMAIN RANGE |
|---|---|---|---|
| CHILD | 20 | 25 | 2-6 |
| INSURANCE | 27 | 52 | 2-5 |
| ALARM | 37 | 46 | 2-4 |
| CHILD10 | 200 | 257 | 2-6 |
| INSURANCE10 | 270 | 556 | 2-5 |
| ALARM10 | 370 | 570 | 2-4 |
| GENE | 801 | 972 | 3-5 |

Table 1: Networks used in experiments and their properties. The domain range is the range of the domain sizes of the variables in the network (e.g a domain range of 2-3 indicates binary variables and variables with 3 possible states).

Note that the suffix 10 means the original network was pasted 10 times and connected randomly as in [31]. All experiments are performed using 1000 data samples except where noted. We found that using more data samples leads to over-fitting, where networks with a higher score than the network which generated the data are found. Also, some real-world data sets may not contain an immense amount of data samples, so 1000 samples is a balance between over-fitting and having enough information about the distribution.

Unless noted in a specific section, experiments on networks with under 100 variables are learned with 10-sample MBS with 20 Monte Carlo and 10 hill climbing steps per iteration. Networks with over 100 variables are learned with 15-sample MBS with 40 Monte Carlo and 15 hill climbing steps per iteration. Sparse candidate experiments use maximum parents parameter of $k = 5$ and conditional mutual information as the statistical test. MMHC, Hill Climbing, and MBS terminate after 15 iterations with no improvement.

### 5.1 Experiments with MBS

#### 5.1.1 Variance

Since MBS is a stochastic search algorithm, we show the results of running MBS on the same data with different random seeds. Results (Figure 5) indicate a low variance on the output of MBS, and we can see that it performs consistently regardless of the randomization seed. The standard deviation is over 15 runs per data set.

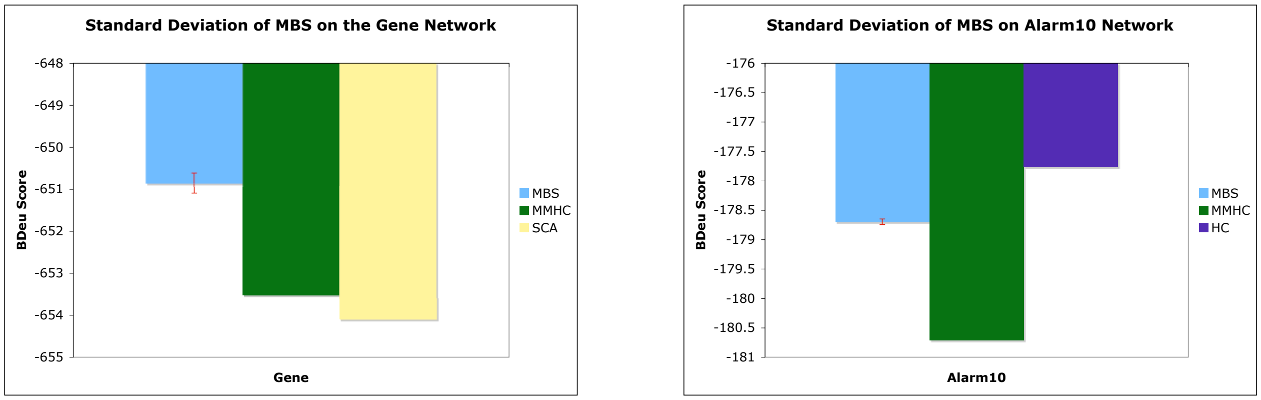


Figure 5: MBS is a stochastic algorithm and its variance is shown on data sets Gene and Alarm10. The error bars indicate one standard deviation. The graphs show that the variance of MBS is small enough that we can consistently compare results with the other deterministic algorithms like hill climbing, MMHC, and SCA.

### 5.1.2 The Model

To show the benefit of maintaining a model of the search space, we ran 15 independent trials of 1-sample MBS and compared the results with a 15-sample MBS run. 15 runs of 1-sample MBS is a distributed search that does not use any information during search since the samples do not communicate. The results are shown in Figure 6. None of the fifteen trials reached the score that 15-sample MBS obtained in the equivalent number of iterations. 15-sample MBS also maintains the highest score throughout each iteration. This makes clear the advantage of using the knowledge gained in search to bias the next iteration.

### 5.1.3 Distance Metric

We performed experiments using four different distance metrics. The first is setting the distance between every sample as 1. The second is the exclusive-or of the two graphs. The third is the exclusive-or discounted by reversed edges and the fourth is the third metric weighted by mutual information. These experiments were run on the child10 data set and the results are shown in table 2. Four experiments per trial and 10 trials were performed. Every trial had a unique random seed that was shared by the four different distance metric experiments per trial. The results show that the first metric, having every sample equidistant from the others, performs the worst. The simple exclusive-or is the second worst of the metrics. Unfortunately the metric weighted by mutual information did not outperform the third metric as we had hoped. This leaves future work to determine if the metric can be improved, perhaps by a weighting or a different statistical test. The results do indicate that the distance metric affects performance, and our intuition about improving the distance metric is correct.

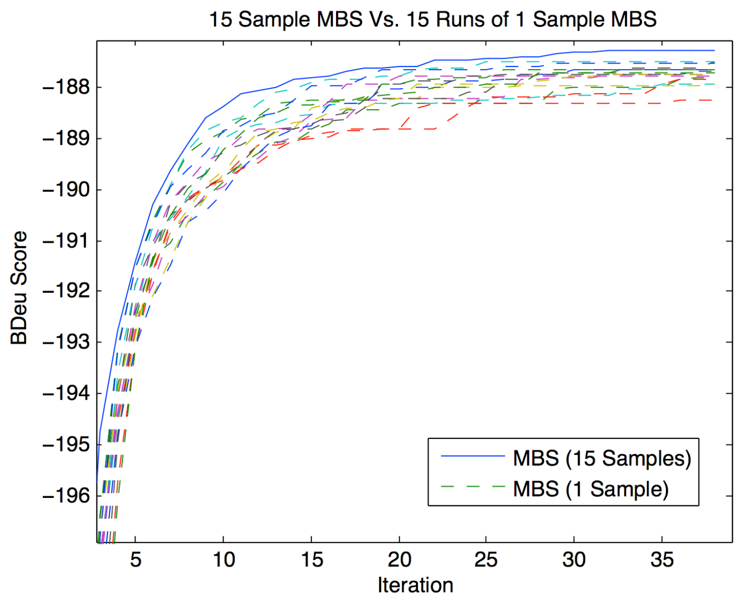


Figure 6: The performance of 15 independent 1-sample MBS runs represented by dashed lines versus a single run of 15-sample MBS. MBS with 15 samples shares information about the landscape and achieves better results than multiple, independent runs of 1-sample MBS, which can be thought of as a generic stochastic search.

| | Metric 1 | Metric 2 | Metric 3 | Metric 4 |
|---|---|---|---|---|
| Average Score | -187.242866 | -187.235452 | -187.216303 | -187.220316 |
| # Best Score | 0 | 2 | 5 | 3 |

Table 2: The average score for each metric and the number of times it scored best over 10 trials

### 5.1.4 Number of Samples

The results varying the amount of MBS samples are in Figure 7. The graph is expected to look like a decaying exponential since at some point adding more samples will not significantly help exploration of the search space. Since the experiments are not repeated, the variance of the algorithm is not accounted for and the graph has some outliers. The graph does indicate that 10-sample MBS performs best on this particular trial of the Gene network. Since using hill climbing in sample improvement accounts for the majority of the score increase, even 5-sample MBS achieves good results. So if computation time is an issue, using low-sample MBS will not dramatically detract from the results.

### 5.1.5 Sample Improvement

By using different sample improvement parameters we can achieve different results and different properties for our final networks. The experiments were run on the Insurance10 data set with 15 samples. The results of running different sample improvement parameters are presented in Table 3 and are sorted by the ratio of Monte Carlo steps to Hill Climbing steps per iteration of MBS. We can see the best results considering score and SHD is a 2 to 1 Monte Carlo to Hill Climbing ratio.

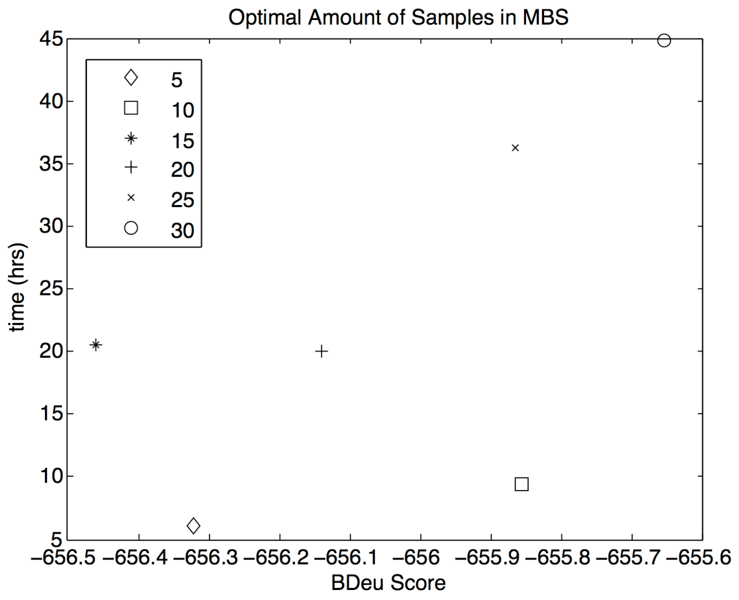


Figure 7: The performance of MBS on the Gene network with varying amounts of samples. The optimal setting would be at the bottom right (minimum time and maximum score). Without the outliers, the graph shows that the performance of MBS levels off when a large number of samples is used, and the increase in score comes at a very high computational cost.

| MONTE CARLO STEPS | HILL CLIMBING STEPS | BDEU SCORE | SHD |
|---|---|---|---|
| 20 | 20 | -224.9327 | 305 |
| 15 | 10 | -225.6822 | 307 |
| 30 | 20 | -224.2427 | 281 |
| 30 | 15 | -224.0934 | 272 |
| 20 | 10 | -224.2876 | 281 |
| 40 | 15 | -224.1827 | 279 |
| 15 | 5 | -224.8624 | 297 |
| 10 | 3 | -225.0196 | 284 |
| 40 | 10 | -224.7923 | 276 |
| 80 | 20 | -223.9007 | 279 |
| 5 | 1 | -225.5447 | 308 |
| 100 | 5 | -224.7261 | 308 |

Table 3: MBS with different ratios of Monte Carlo steps to Hill Climbing steps. Below the middle line the ratio of Monte Carlo to Hill Climbing steps is less than 2 to 1.

## 5.2 Score Improvement

Figure 8 shows the the BDeu score achieved on the low dimensional networks over 10 data sets (referred to as trials) per network . Model-Based Search beats or equals hill climbing in the MMPC search space (MMHC) in each run and occasionally beats regular hill climbing, which searches the entire space of legal moves. Hill climbing is the leader on low-dimensional networks because the search space is not very complex, and there is essentially a single large hill in the landscape that hill climbing can easily scale. MBS can only match or slightly improve on MMHC when the reduced search space does not contain all the edges of the optimal network.

When we consider the high dimensional networks in Figure 9, MBS shows even more improvement over MMHC. On the gene network data, MBS scores higher than MMHC on all runs and Sparse Candidate on all but one run. MBS clearly outperforms hill climbing when running in the same search space since it is more complex and hill climbing gets stuck at a lower maximum. MBS is also the best performer on the network with the highest dimensionality, Gene.

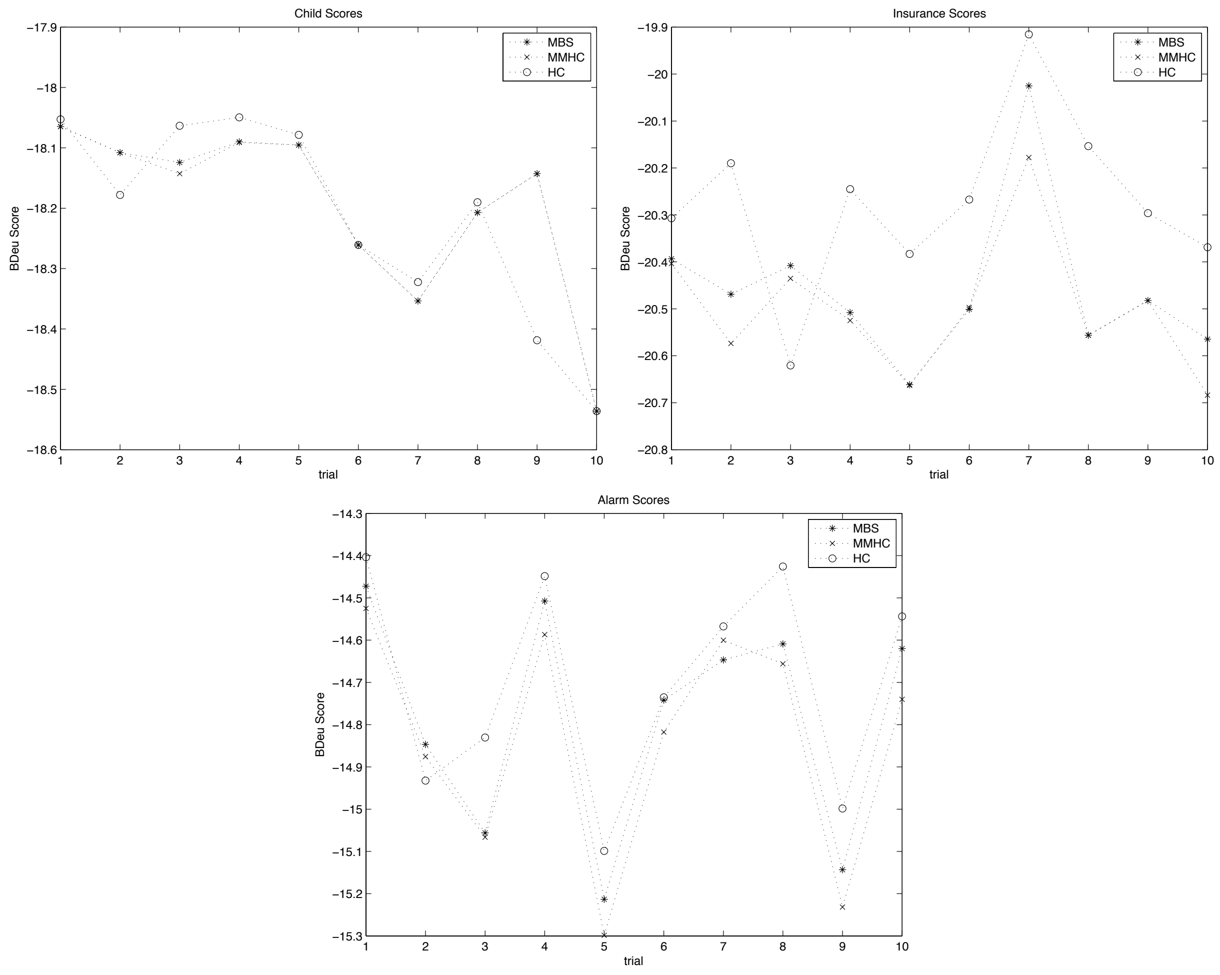


Figure 8: Score results for low dimensional networks Child, Insurance, and Alarm for 10 data sets generated per network

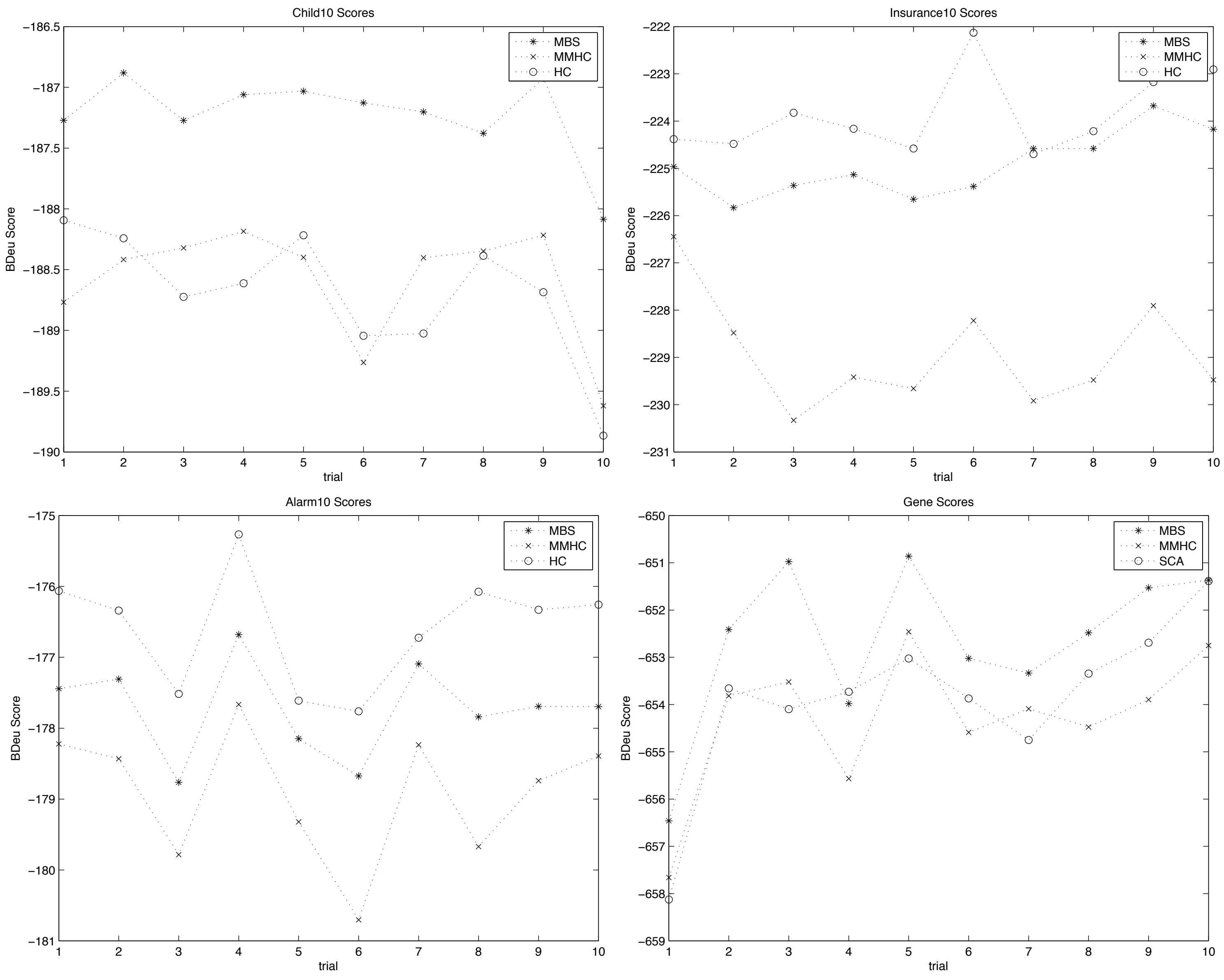


Figure 9: Score results for high-dimensional networks Child10, Insurance10, and Alarm10 and Gene for 10 data sets generated per network.

## 5.3 Structural Hamming Distance

Although optimizing the BDeu score is an efficient method of reconstructing a Bayesian network, it is flawed as a metric of network reconstruction. The score only corresponds to the a posteriori probability of a network under certain assumptions, which the data may or may not uphold. Also, the score is dependent on the amount of data samples and the particular priors used, so comparing results on varying amounts of data is meaningless. It is also important to compare the learned network to the network which created the data. The Structural Hamming Distance (SHD) [31] is a similarity metric between two PDAGs. Since Bayesian networks can be statistically equivalent yet structurally different, we would like to compare two networks in their equivalent class form as a PDAG. All the algorithms return DAGs instead of PDAGs, so we convert DAGs to their equivalent PDAG using the method presented in [4]. The SHD algorithm is implemented as described in [31].

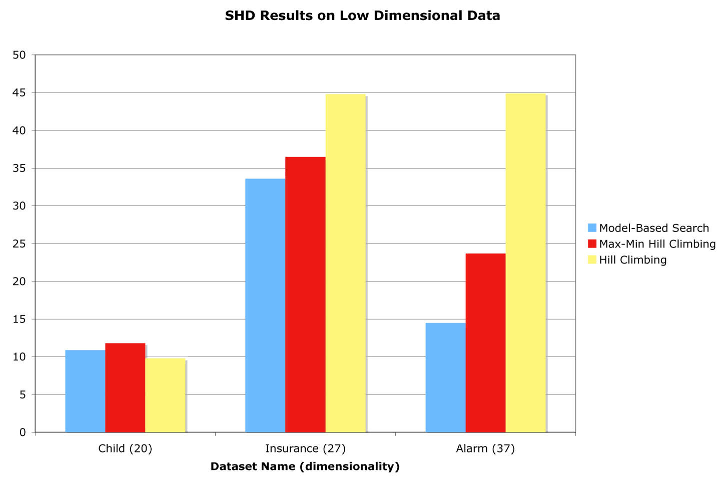


Figure 10: SHD results on low dimensional networks

The SHD results indicate the distance between the PDAG of the network encountered by search and the PDAG of the optimal network (Figures 10, 11 and 12). The SHD is averaged over ten data sets that were generated from each network. A lower SHD indicates similar PDAGs. The graphs show that MBS consistently returns structures that are structurally the most similar to the network which generated the data. Hill climbing performs worse because it will keep adding edges of low statistical significance if they improve the score. MBS and MMHC use a constrained search space which inherently prevents the methods from adding the spurious edges that are outside of the MMPC search space.

## 6 Conclusion and Future Work

Structure learning of Bayesian networks is a computationally expensive task that requires innovative algorithms to produce high quality results. When high-dimensional data sets are learned, traditional search and score methods cannot scale to handle the increase in search space size. Reducing the search space using heuristics and running search has produced significant results, but there remains room for improvement. This paper applies Model-Based Search, a global search technique, in a reduced search space. The results are higher quality structures on high-dimensional networks than other leading structure learning algorithms.

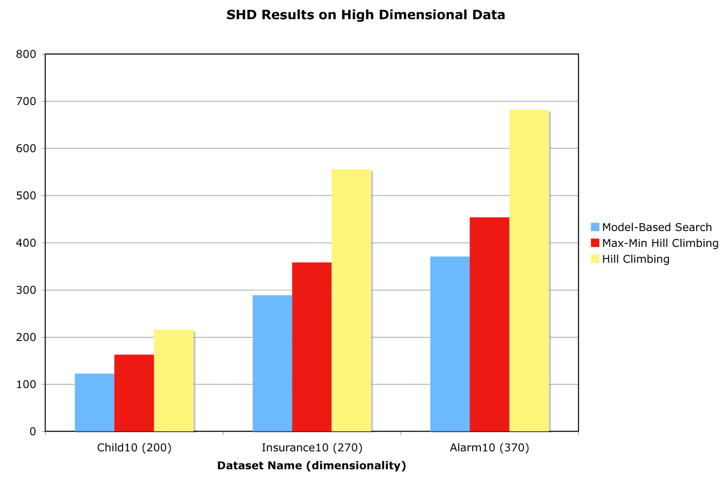


Figure 11: SHD results on high-dimensional networks

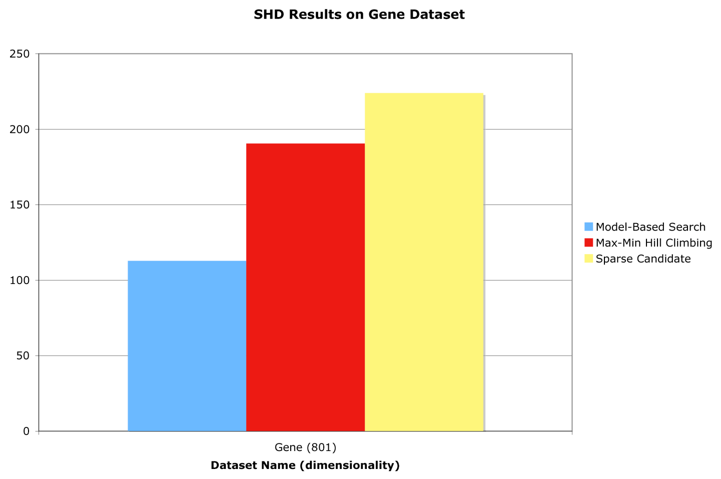


Figure 12: SHD results on the Gene network

The search achieves an improvement in the BDeu score and also in the Structural Hamming Distance indicating structures that are more similar to the "optimal" structure which generated the data. The paper also includes experiments on the performance of Model-Based Search. We show it is a low variance, stochastic algorithm, which means it produces similar results consistently. Experiments also show the benefits of maintaining a model of the search space. Finally experiments varying the parameters in MBS indicate optimal settings and provide inspiration for future work.

Future work is abundant in applying Model-Based Search to structure learning. The author would like to run MBS on more high-dimensional data sets and also obtain performance results on real-world data sets. Finding the optimal settings for sample improvement is also a challenge, as is exploring different sampling methods. Finally, implementing a high performance variation of MBS mentioned in section 4.3

can potentially produce even better results than obtained by the current version.

## 7 Acknowledgments

The authors would like to thank TJ Brunette for help with Model-Based Search and suggestions on the implementation and this paper. We would also like to thank Sridhar Mahadevan for his additional review of this paper. Last, special thanks goes to Laura Brown of the Discovery Systems Laboratory for help with MMHC and Causal Explorer.

## References

[1] TJ Brunette and Oliver Brock. Improving protein structure prediction with model-based search. *Bioinformatics*, 21:66–74, 2005.

[2] Jie Cheng, Russell Greiner, Jonathan Kelly, David Bell, and Weiru Liu. Learning Bayesian networks from data: an information-theory based approach. *Artificial Intelligence*, 137(1-2):43–90, 2002.

[3] D. M. Chickering, D. Geiger, , and D. Heckerman. Learning Bayesian networks: Search methods and experimental results. In *Proceedings of the Fifth Conference on Artificial Intelligence and Statistics*, pages 112–128, Ft. Lauderdale, FL, 1995.

[4] David Chickering. A transformational characterization of equivalent Bayesian network structures. In *Proceedings of the 11th Annual Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pages 87–98, San Francisco, CA, 1995. Morgan Kaufmann.

[5] David Chickering, Christopher Meek, and David Heckerman. Large-sample learning of Bayesian networks is np-hard. In *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence*, pages 124–133, San Francisco, CA, 2003. Morgan Kaufmann Publishers.

[6] David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2003.

[7] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

[8] Nir Friedman and Daphne Koller. Being Bayesian about network structure. a Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50(1-2), January 2003.

[9] Nir Friedman, Iftach Nachman, and Dana Pe'er. Learning Bayesian network structure from massive datasets: The "sparse candidate" algorithm. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence*, pages 206–215, San Francisco, CA, 1999. Morgan Kaufmann Publishers.

[10] Zoubin Ghahramani and Michael I. Jordan. Supervised learning from incomplete data via an em approach. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, pages 120–127, 1993.

[11] Paolo Giudici and Robert Castelo. Improving markov chain monte carlo model search for data mining. *Machine Learning*, 50(1-2):127–158, 2003.

[12] Fred Glover and Fred Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[13] Anna Goldenberg and Andrew Moore. Tractable learning of large bayes net structures from sparse data. In *Proceedings of the Twenty-First International Conference on Machine Learning*, page 44, New York, NY, USA, 2004. ACM Press.

[14] David Heckerman, Dan Geiger, and David M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

[15] Jaime S. Ide and Fabio Gagliardi Cozman. Random generation of Bayesian networks. In *SBIA '02: Proceedings of the 16th Brazilian Symposium on Artificial Intelligence*, pages 366–375, London, UK, 2002. Springer-Verlag.

[16] R. Jansen, H. Yu, D. Greenbaum, Y. Kluger, N. J. Krogan, S. Chung, A. Emili, M. Snyder, J. F. Greenblatt, and M. Gerstein. A Bayesian networks approach for predicting protein-protein interactions from genomic data. *Science*, 302(5644):449–453, 2003.

[17] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[18] Mikko Koivisto and Kismat Sood. Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, 5:549–573, 2004.

[19] W. Lam and F. Bacchus. Learning Bayesian belief networks: An approach based on the mdl principle. *Computational Intelligence*, 10(3):269–293, 1994.

[20] P. Larrañaga, M. Poza, Y. Yurramendi, R. H. Murgam, and C. M. H. Kuijpers. Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(9):912–926, 1996.

[21] D. Madigan and J. York. Bayesian graphical models for discrete data. *International Statistical Review*, 63:215–232, 1995.

[22] N. Metropolis and S. Ulam. The monte carlo method. *Journal of the American Statistical Association*, 1949.

[23] Andrew Moore and Weng-Keen Wong. Optimal reinsertion: A new search operator for accelerated and more accurate Bayesian network structure learning. In T. Fawcett and N. Mishra, editors, *Proceedings of the 20th International Conference on Machine Learning*, pages 552–559, Menlo Park, California, August 2003. AAAI Press.

[24] Jens Nielsen, Tomas Kocka, and Jose Peña. On local optima in learning Bayesian networks. In *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence*, pages 435–442, San Francisco, CA, 2003. Morgan Kaufmann Publishers.

[25] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman Publishers, 1988.

[26] Judea Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–709, 1995.

[27] R. W. Robinson. Counting labeled acyclic digraph. In F. Harary, editor, *New Directions in Graph Theory*, pages 239–273. New York: Academic Press, 1973.

[28] P. Spirtes, C. Glymour, and R. Scheines. Causality from probability. In *Proceedings of Advanced Computing for the Social Sciences*, Williamsburgh, VA, 1990.

[29] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. The MIT Press, 2000.

[30] I. Tsamardinos, C. Aliferis, and A. Statnikov. Algorithms for large scale markov blanket discovery, 2003.

[31] I. Tsamardinos, L.E. Brown, and C.F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 2006.