A Chain-Model Genetic Algorithm for Bayesian Network Structure Learning

Ratiba Kabli
School of Computing
The Robert Gordon University
Aberdeen, United Kingdom
rk@comp.rgu.ac.uk

Frank Herrmann
School of Computing
The Robert Gordon University
Aberdeen, United Kingdom
fh@comp.rgu.ac.uk

John McCall
School of Computing
The Robert Gordon University
Aberdeen, United Kingdom
jm@comp.rgu.ac.uk

ABSTRACT

Bayesian Networks are today used in various fields and domains due to their inherent ability to deal with uncertainty. Learning Bayesian Networks, however is an NP-Hard task [7]. The super exponential growth of the number of possible networks given the number of factors in the studied problem domain has meant that more often, approximate and heuristic rather than exact methods are used. In this paper, a novel genetic algorithm approach for reducing the complexity of Bayesian network structure discovery is presented. We propose a method that uses chain structures as a model for Bayesian networks that can be constructed from given node orderings. The chain model is used to evolve a small number of orderings which are then injected into a greedy search phase which searches for an optimal structure. We present a series of experiments that show a significant reduction can be made in computational cost although with some penalty in success rate.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search - *heuristic methods*; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Performance, Experimentation

Keywords

Bayesian Networks, Genetic Algorithms, Greedy Search

1. INTRODUCTION

Bayesian Networks (BNs) are probabilistic models useful for reasoning with, or representing knowledge under uncertainty. Essentially a BN can be defined as a pair (G, P) where G is a directed acyclic graph (DAG) $G = (V, E)$ with the vertices V as the nodes in the network. Each node represents a random variable X_i relevant to the problem domain. The dependencies among these variables are represented by the set of edges E in the underlying DAG factorizing the joint probability distribution $P(X)$ over the set of random variables X_n into a conditionally independent one

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i|Pa(X_i)) \tag{1}$$

with $Pa(X_i)$ as the set of parent nodes for node X_i. To make use of the power of Bayesian Networks in knowledge representation and inference, they have to first be constructed for the given problem. To fully specify a Bayesian network, one has to first define the underlying DAG structure representing the network and then the BN's conditional probability distribution. For the latter, we have to determine the prior probabilities for all the nodes with no incoming links (i.e. root nodes), and all conditional probabilities for the remaining nodes. This is essentially a parameter estimation problem and is not discussed in the present work.
In this paper, we look at the structure learning part of specifying Bayesian networks. Although time consuming and not always possible, this can be achieved manually given extensive knowledge of the domain from experts or available literature. A popular alternative that has emerged, however, from within the artificial intelligence community in recent years, is the attempts to empirically induce BN structures from data. Various algorithms have in fact been developed in order to deal with this problem. Nevertheless, even when an array of assumptions and restrictions are made, structure learning for Bayesian Networks remains a hard problem [8]. This is mainly due to the fact that the number of possible structures for a given problem grows super exponentially given the number of variables in that problem. Robinson [24] quantifies the number as $O(n!2^{\binom{n}{2}})$ for a problem of size n. So where a 3 variable problem would have 25 possible networks, a 5 variable problem would have 29,281 and a 6 variable problem would have 3,781,503 possible networks. This makes exact methods for structure discovery impractical and seldom used without imposing a great deal of restrictions [17]. More often used nowadays are approximate methods and approaches based on some heuristics. Mainly, these can be classified into two main groups: Search and Score methods and Conditional Independence Testing methods. The latter is a constraint based approach which relies on a number of statistical tests to determine whether two variables are independent or dependent given a set of conditioned variables. Tests such as Pearson's Chi-Square

and mutual information are often used. Work by de Campos [12] and Spirtes and Glymour's PC algorithm [25] illustrate this. This approach tends to give good results with sparse networks and small samples of data however it does not scale very well for large datasets and dense networks.

In this work, we are interested in the Search and Score approach and more specifically in the use of genetic algorithms and greedy search in inducing Bayesian network structures from data. A brief overview of work done based on this approach follows in the next section. We then look at research done in learning Bayesian network structures by searching through the space of orderings in which we are particularly interested for this work. In Section 4 we describe our approach and hypothesis with regards to the usefulness of using chain structures in evaluating node orderings. We then describe the experiments carried out for testing this hypothesis in Section 5. We discuss the results in Section 6 and conclude in Section 7.

2. SEARCH AND SCORE METHODS FOR LEARNING BAYESIAN NETWORKS

Bayesian network structure discovery generally involves searching through the space of all possible network structures for one that best describes the data. Traditionally, this is done by employing some search mechanism along with an information criterion to measure goodness and differentiate between candidate structures met while traversing the search space. The idea would be to try and maximize this information measure or score by moving from one structure to another by means of some local variation such as a deletion or an addition of a link between two nodes and then evaluating the overall effect of the move. This is iterated until an optimal score is found. The associated structure is then chosen to represent and explain the data. There has been a lot of work done in both score functions and search algorithms used for this purpose. Examples of scoring metrics include the AIC metric, BDeu, the Bayesian metric, the CH metric, the Minimum Description Length (MDL), etc. An overview of these metrics can be found in [4, 10, 11, 16]. On the other hand, there has also been a great deal of variation in terms of the search strategies used. Some of these methods such as a hill climber by Buntine [6] look at a single network structure at a time and iteratively modify it until a good solution or some stopping criteria has been reached. Examples of this approach include Boukaert's application of Simulated Annealing [5], Goldzmit and Friedman's TAN method [13] based on work done by Liu and Chow [9], the B algorithm and popular greedy search algorithm K2 developed by Cooper and Herskovits [10]. We will review the latter in more details in the next sections. Contrarily to the approach above, there also has been research into methods which consider a group of network structures at a time rather than a single structure. They also use a scoring function to evaluate the current group (or population of structures) from which a new and better population is created and then evaluated and so on. The algorithms iterate until a set stopping criteria is reached. The advantage of this approach is that it tends to explore the search space better and therefore has lower chances of getting stuck in local optima. Various algorithms which follow this approach have been proposed. They use evolutionary algorithms such as genetic programming [29], genetic algorithms [15, 18, 19, 22, 27]. Other approaches to structure learning of Bayesian networks look at another aspect of the learning process; mainly the search space. They focus on ways to reduce this space and thus making any search strategy more tractable. Some contribution to this problem include work by Friedman [14] with the sparse candidate algorithm, Provan [23] with feature/attribute selection, searching over the space of equivalence classes, and discussed in this paper ordering based searches [18, 26]. In recent years, work on using Markov Chain Monte Carlo methods and other related techniques should also be noted [30].

3. SEARCHING THROUGH THE SPACE OF NODE ORDERINGS

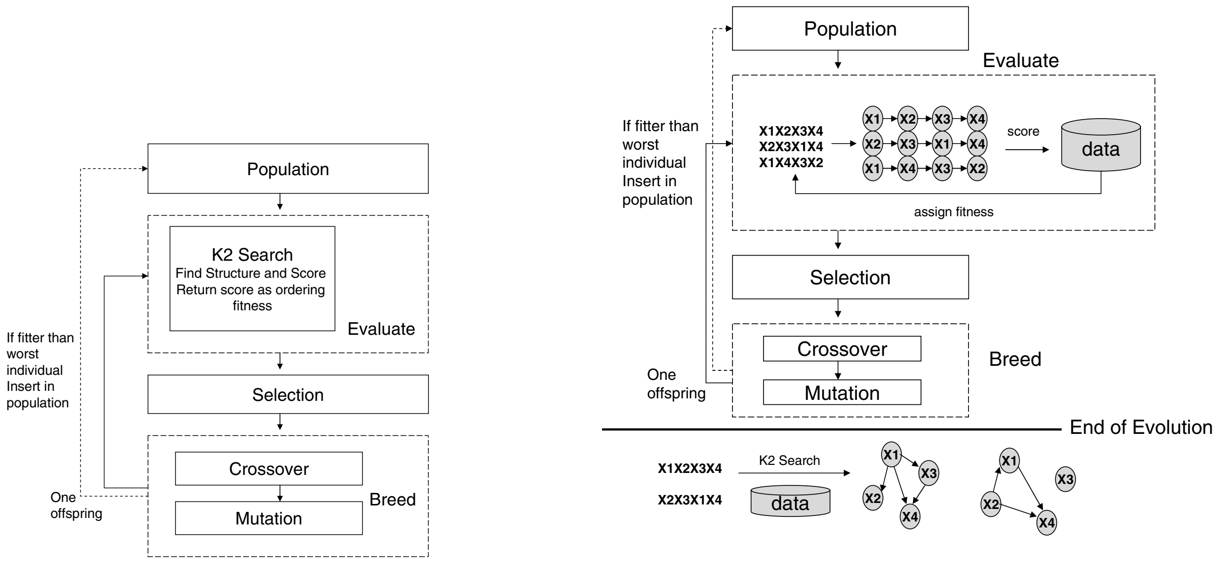
The nodes in a Bayesian network admit at least one ordering based on the DAG structure underlying the network. This topological ordering imposes the property that a node X_i is dependent on X_j implies $X_i < X_j$ in the ordering. In other words, a node can only be a parent to another node if it precedes it in the node ordering. One way of searching for Bayesian network structures is to search through this space of orderings, looking for those that will admit good structures. This is more efficient than searching through the space of structures as it reduces the search further eliminating all cyclic structures and structures incompatible with the given ordering. It remains to say however that an exhaustive search through all orderings for large problems remains intractable ($n!$ for a problem of size n), and therefore heuristics are generally used.

In [18], Larrañaga et al. propose a genetic algorithm to search the space of node orderings rather than the full space of structures. For the purpose of this paper, we denote this algorithm by K2GA. The initial individuals in the population are randomly created node orderings which are then evolved until a good ordering is found. At each iteration, two individuals are selected for crossover and mutation given the rank of the value of their fitness in the population. Only one individual offspring is created at a time and in case is better, it replaces the worst individual in the current population. Figure 1a illustrates this process. The fitness of each ordering is calculated by running the greedy search algorithm on that ordering at each time and returning the score of the network structure found.

The greedy search algorithm used is the K2 proposed by Cooper and Herskovitz [10]. The algorithm assumes that a priori, all structures are equally likely and that cases in the data occur independently and are complete. Moreover, it assumes the presence of a node ordering and imposes a maximum number of parents a node can have (inbound edges). With these conditions satisfied, K2 starts with an empty ancestor set for each node and incrementally adds links that maximize the score of the resulting structure. The algorithm stops when no more ancestor node additions improve the score. K2 was originally used along the CH score which captures the probability of a candidate network structure B_s given a set of data D. Formally the discrete probability $P(B_s, D)$ is given by

$$P\left(B_s, D\right) = P\left(B_s\right) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \qquad (2)$$

Where q_i denotes the number of possible different instances



a) K2GA b) Chain-Model GA

Figure 1: Genetic Algorithm for Bayesian Network Structure Learning

the parent of variable X_i can take. r_i is the number of values X_i has, N_{ijk} denotes the number of cases in the dataset D in which X_i takes value k of its x_i instance when its parent Pa_i has its j^{th} value. N_{ij} is the sum of all N_{ijk} for all values x_i can take. Although simple to implement and widely used, K2 is prone to local optima and may not find the globally best structure. Moreover, it relies on prior knowledge of the node ordering and does not guarantee to find an equivalent network structure given any ordering. Computationally also, K2 can be very expensive. A cheaper alteration we can make to K2 would be to use a less expensive scoring metric such as the MDL score based on the Minimum Description Length Principle [4]. The latter is based on entropy and states that the description length of a Bayesian network and a dataset of cases is the sum of the size of the network and the size of the dataset after it has been compressed using that network. The best network is thus the one with the smallest description length. MDL balances complexity and accuracy of the resulting networks. Wong [29] and van Dijk [27] both use MDL as a measure of goodness in learning Bayesian networks. Although cheaper compared to the CH metric, we still face the problem of running K2 search to evaluate each ordering in our evolution process. In this paper, we look at ways of reducing the computational cost related to this.

4. FILTERING GOOD ORDERINGS

4.1 Chain Structures

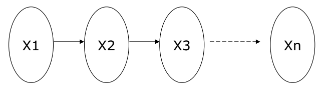
We investigate the use of chain structures to evaluate orderings by replacing the K2 expensive evaluation in Larrañaga's genetic algorithm. We base our approach on a hypothesis that a chain is a sufficiently good model to locate node orderings of which good Bayesian network structures can be built. In Section 5 we describe the experiments we ran in order to test this hypothesis.

We make the orderings' evaluation step cheaper by evaluating their associated *chain structure* rather than search for the best structure with K2. In other words, given a set of variables $(X_1, X_2, ..., X_n)$ and an ordering $X_1, X_2, ..., X_n$, then it admits a chain structure $X_1 \rightarrow X_2 \rightarrow, ..., \rightarrow X_n$ i.e. X_i is the sole parent of X_{i+1} (Figure 2a).

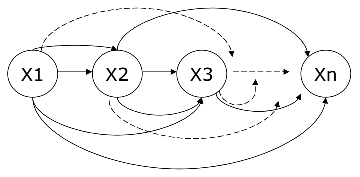
We hypothesise that the ordering dominates the score for a given network structure and therefore more time should be spent searching for a good structure given a good enough ordering. In other words, we evaluate the node ordering by building its associated chain structure (the structure resulting from drawing a link between each node and the node proceeding it in the node ordering) and evaluating it given the data with our scoring metric. We hypothesise that this chain structure, although simple, may hold important information about the ordering and its relation to the network structure. This is, intuitively true for many practical domains where the links between the nodes represent genuine causal relationships. For instance, medical data usually follows a diagnosis, treatment and outcome pattern.

4.2 Chain-Model GA

We now define the proposed Chain-Model GA, illustrated in Figure 1b. Following the same GA approach used by K2GA, the ChainGA as we will refer to our Chain-Model GA, differs in its evaluation step, where a chain structure of the given ordering is evaluated and the ordering assumes the fitness returned by the chain. This low resolution evaluation phase preselects orderings, rejecting all not good enough ones and keeping the ones that could be good ones. The ChainGA model then adds on another step at the end of



a) Chain Structure



b) Complete Structure

Figure 2: Proposed Structures

Table 1: The Genetic Algorithm Settings

GA Settings	**Asia Dataset**	**Car Dataset**	**Alarm Dataset**
Evolution type	Steady State (Genitor)		
Number of Offspring	1		
Selection	Rank Selection		
Crossover	Cycle Crossover: rate:0.9		
Mutation	Displacement Mutation: rate:0.1		
Population size	100	20	10

each evolution where K2 is run on a percentage of the best orderings to search for a good structure. This reduces computation time since the number of links to evaluate are fixed in contrast to K2. However, there is a need to investigate this and the quality of the resulting network structures.

In order to evaluate our method discussed above, we ran a series of experiments on three benchmark datasets. For comparisons, we implemented the K2GA algorithm. Netica was used for data sampling and BNJ [3] was used to view the resulting network structures saved in XML format. For the implementation of our Chain-Model GA approach, we chose to keep the choice of evolution described in Larrañaga's work, with a steady state like evolution mimicking the Genitor system developed by Whitley [28]. The crossover and mutation operators used; respectively cycle crossover with a rate of 0.9 and displacement mutation with rate of 0.1 were also kept as they performed best with node ordering individuals. Table 1 summarizes the evolution parameters used for all three sets of experiments. The difference is in the evaluation function.
In our algorithm, to evaluate each node ordering, we build its chain structure and score it given the data, with the CH metric used by K2. At the end of each run, we run K2 on a group of orderings in the current population and record the resulting structures and their scores. We have experimented with several options in respect of the orderings to run K2 on. Mainly, on the best ordering in the current population, a random selection of orderings and a group of the best orderings in the population. Work done has shown that choosing a group of the best orderings resulted in the best performance. We have chosen to run the K2 full search on 10 to 50% of the individuals in the population, depending on the population size used. At the end of the run, the group of structures and choices generated by K2 are compared and the ordering corresponding to the best scoring network structure is selected as the best individual of the run. It should be noted that we kept the inbound degree, or the number of ancestor nodes permitted for each node to four.

5. EXPERIMENTAL RESULTS

We ran the algorithms several times and for each, we recorded: the best structure found, its associated score and the number of factor evaluations done. In this paper, we define a factor evaluation as being the count of times the term F in Formula 3 is accessed when Formula 2 is used. Since the score can be decomposable in its log form, it is possible to retrieve a node and its family score from the overall network score. In other words, we count as factor evaluations the number of times operations are needed to build and score the parent set for a specific node X_i in F

$$F = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \qquad (3)$$

We can already anticipate less factor evaluations by this approach since for the simple chain structures the parent set for any node is fixed given the ordering that generated that structure. Therefore, for each ordering of size n, a fixed n factor evaluations for its chain structure are performed. In contrast to the number of factor evaluations needed by K2 search. However, we should note that the number of evaluations recorded for the ChainGA approach also includes the number of evaluations done by the K2 phase carried out at the end of each run.

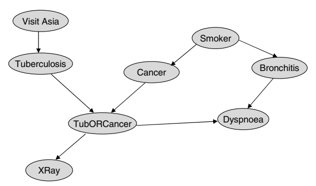


Figure 3: Original Asia Network Structure

5.1 The Asia Network

We first evaluated our approach on the Asia (Chest Clinic) benchmark dataset. A diagnostic demonstrative Bayesian network, it first appeared in work done by Lauritzen and Spiegelhalter [20]. The Asia network (Figure 3) is a simple network with 8 binary nodes and 8 edges and allows validation in a straightforward manner. We generated a dataset of 5000 cases for the Asia network which we will use as our learning dataset. The cases were simulated using Netica's case simulation facility [21].

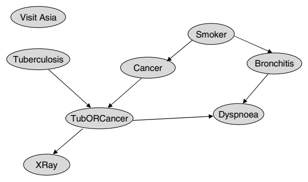


Figure 4: Recovered Asia Network Structure

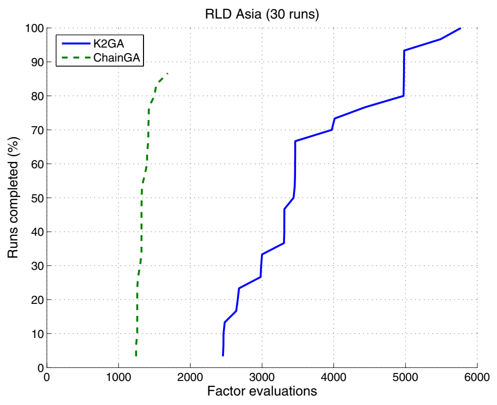


Figure 5: Run Length Distribution (Asia Network 30 Runs)

5.2 The Car Diagnosis Network

The Car Diagnostic Network is another artificial network present in different versions in most of the available Bayesian network tools e.g. in Hugin, Netica, BayesiaLab [1]. It consists of 18 nodes and 17 edges (see Figure 6). We used BayesiaLab's version of the network and generated a dataset of 10000 cases for the purpose of the experiment. Table 3b shows the results for 50 runs of each algorithm.

5.3 The Alarm Network

We also evaluate the two approaches on the Alarm monitoring network dataset [2]. The Alarm network is a medical diagnostic system for intensive care patient monitoring and consists of 37 nodes and 46 edges. For these experiments, we used a sample of 3000 data cases, sampled using the Netica tool.

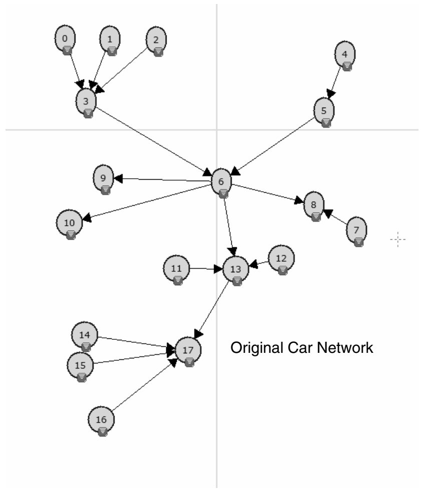
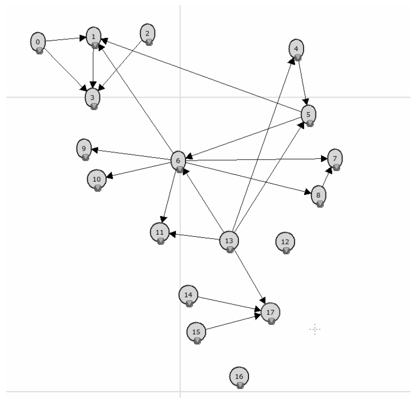
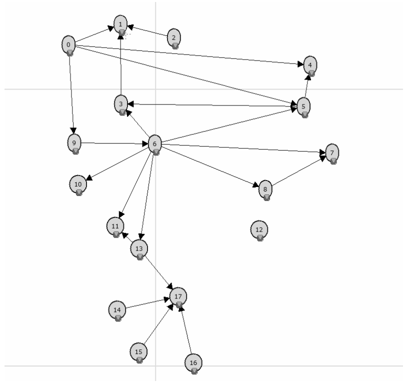


Figure 6: Original Car Network Structure

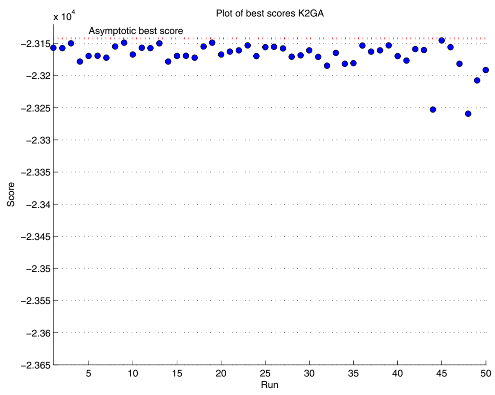


a) K2GA

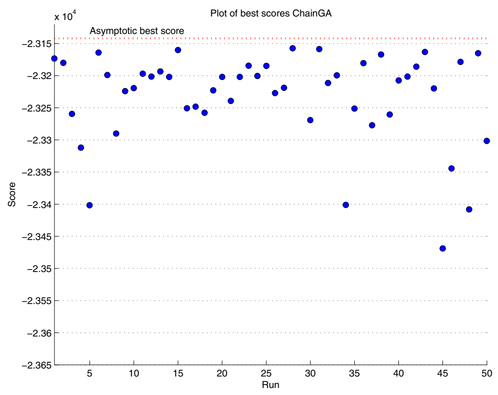


b) ChainGA

Figure 7: Car Network Recovered Structures



a) K2GA



b) ChainGA

Figure 8: Best found Structures at Each Run (Car Network)

Table 2: Success Rates

	Asia Network	**Car Network**	**Alarm Network**
K2GA	100%	98%	75%
ChainGA	90%	%58	75%

Table 3: Experimental Results

	a) Asia 5000 Cases		**b) Car Diagnosis 10000 Cases**		**c) Alarm 3000 Cases**	
	Avg Best Score	**FE's**	**Avg Best Score**	**FE's**	**Avg Best Score**	**FE's**
K2GA	-11244.3 ± 2.0	3645 ± 968	-23168.5 ± 21.0	2227.96 ± 395	-30068.4 ± 164	2458.4 ± 474
ChainGA	-11248.1 ± 11.0	1924 ± 273	-23213.3 ± 152	1018.7.± 177	-30097.3 ± 141	1844(1421) ± 116

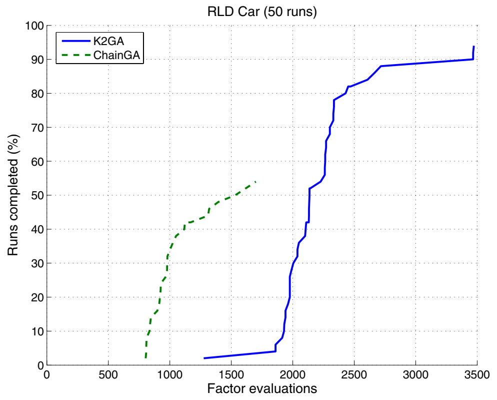


Figure 9: Run Length Distribution (Car Network 50 Runs)

We ran each algorithm 60 times and recorded the score and evaluations results in Table 3c. In this experiment, we fix the number of orderings to go through to the K2 phase to 50% of the population, i.e. 5 orderings in this case.

6. DISCUSSION

For the Asia dataset, an ensemble of 30 experiments were ran for each algorithm. We scored the original network structures and stopped each run when the resulting score is a percentage away from the original network score. As we can see from Figure 4, both algorithms managed to recover a near optimal Asia network structure with one edge missing from the VisitAsia node to Tuberculosis. Table 3a also shows comparable average network scores and success rates (Table 2). The difference is apparent in the number of factor evaluations needed to achieve these structures (Figure 5). We can notice the same pattern for the Car diagnosis problem, where even though the recovered structures are not identical, they are very similar and close to the original network structure (Figures 7a and 7b). The scores illustrated in Table 3b also do not seem to be significantly different. Figures 8a and 8b show the plot of the resulting best scores of all runs for each of the algorithms. Again the noticeable difference seems to be in the number of factor evaluations carried out by each of the algorithms, although the success rate is about 58% for ChainGA (Table 2) for this problem and therefore we might need to run the experiments twice as often to get the same results (Figure 9). More experiments will be carried out in the future in order to explain this result.

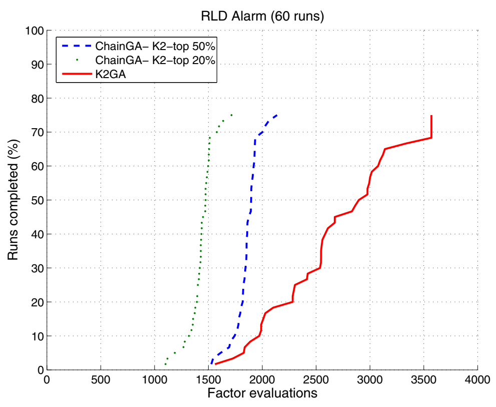


Figure 10: Run Length Distribution (Alarm Network 60 Runs)

For the Alarm problem, we ran each experiment 60 times. Table 3c records the results. The difference between the average best scores achieved by each approach is negligible. However, we can see the reduction in the number of factor evaluations when using the ChainGA approach. Indeed K2GA needs nearly as much as double the number to find similar network structures. Furthermore, the number of evaluations reported in the table is when at the K2 search phase of ChainGA, we run the search on 50% of the population, i.e. in this case 5 orderings. In general we would run only a percentage of 10 to 20% of the population at the end of the run by K2, which would need in this case less number of factor evaluations (e.g.1421 for 20% of the population). Figure 10 shows the run length distribution for each of the algorithms. We can see how the ChainGA runs completed need less factor evaluations than their K2GA counterparts. For both algorithms, the success rate is comparable as we can see from Table 2 about 45 runs have completed successfully out of the 60 runs carried out for each. The best scores found for each run for both algorithms can be seen in Figures 11a and 11b.

We speculate that this good performance is somewhat related to the nature of the problems used. All three networks used in this work are diagnostic and rely immensely on the causal relationship between the factors of the problem. The chain structures emphasize a certain choice of causal links given a certain ordering and therefore where the ordering considered is close to the optimal ordering, the chain structure seem to give us enough information about the network structure to build.

7. CONCLUSION AND FUTURE WORK

We proposed a method for reducing the computation time needed for learning Bayesian network structures. The proposed method uses a chain-model GA which relies on chain structures for the evaluation of node orderings of which Bayesian network structures are then searched for using K2 search. We applied our model to the Asia, the Car and the Alarm networks; three different benchmark datasets of various complexity. We compared our method to the GA designed by Larrañaga for searching for Bayesian network structures through the space of node orderings.

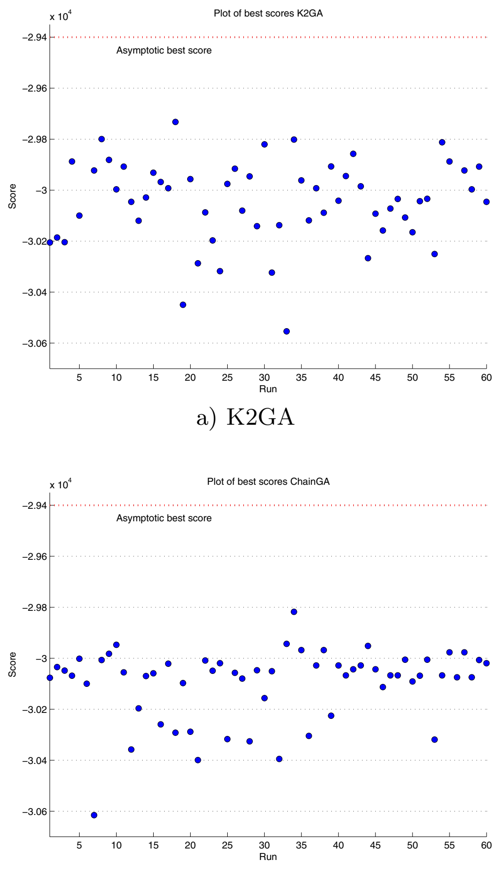


Figure 11: Best found Structures at Each Run (Alarm Network)

Our preliminary results show that our method obtained comparable results in term of structure score and structure topology to those obtained by K2GA but in a reduced number of factor evaluations. This seems promising although for the Car Diagnosis problem as we have shown, the success rate is lower and although this might be solved with restarts, in future work, we would like to investigate the reasons for this and explore whether or not this is problem related. Moreover, to improve success rate we would like to build on the idea of the chain and investigate the complete structure associated with an ordering. I.e. where each node in the ordering is connected to the nodes coming after it in the node order given (Figure 2b). Using this complete structure to filter orderings we speculate will result in better quality network structures as the optimal structure should be contained within this complete structure. In practice this might be less efficient than the simple chain structure approach as it will require a greater number of factor evaluations. Preliminary experiments on the Asia dataset have proved this. Work on customizing the scoring metric we use to cater for these chain structures and not penalize them for complexity is also envisaged. Consequently, another avenue we would like to explore is the use of other available scoring metrics such as the MDL principle.

8. ACKNOWLEDGMENTS

The Authors would like to thank the Robert Gordon University for their support and funding of this research.

9. REFERENCES

[1] Bayesia. Bayesialab Bayesian network software. http://www.bayesia.com/.

[2] I. A. Beinlich, H. J. Suermondt, R. M. Chavez, and G. F. Cooper. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks,. *Proceedings of the Second European Conference on Artificial Intelligence in Medical Care, Springer-Verlag, Berlin.*, pages 247–256, 1989.

[3] BNJ. Bayesian network tools in Java. http://bnj.sourceforge.net/.

[4] R. R. Bouckaert. Probabilistic network construction using the minimum description length principle. *Lecture Notes in Computer Science*, 747:41–48, 1993.

[5] R. R. Bouckaert. *Bayesian Belief Networks: From Construction to Inference*. PhD thesis, University of Utrecht, 1995.

[6] W. Buntine. Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, 2:159–225, 1994.

[7] D. Chickering. Learning Bayesian networks is NP-Complete. In *Proceedings of AI and Statistics*, 1995.

[8] D. Chickering, D. Heckerman, and C. Meek. Large-sample learning of Bayesian networks is NP-Hard. *J. Mach. Learn. Res.*, 5:1287–1330, 2004.

[9] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, 14:462–467, 1968.

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

[11] R. G. Cowell, S. L. Lauritzen, A. P. David, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1999.

[12] L. de Campos and J. Huete. On the use of independence relationships for learning simplified belief networks. *International Journal of Intelligent Systems*, 12:495–522, 1997.

[13] N. Friedman and M. Goldszmidt. Learning Bayesian networks with local structure. *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, pages 252–262, 1996.

[14] N. Friedman, I. Nachman, and D. Peér. Learning Bayesian network structure from massive datasets: The "sparse candidate" algorithm. In *Uncertainty in Artificial Intelligence*, pages 206–215, 1999.

[15] J. Habrant. Structure learning of Bayesian networks from databases by genetic algorithms-application to time series prediction in finance. In *ICEIS*, pages 225–231, 1999.

[16] D. Heckerman, D. Geiger, and D. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. In *KDD Workshop*, pages 85–96, 1994.

[17] M. Koivisto and K. Sood. Exact bayesian structure discovery in Bayesian networks, 2004.

[18] P. Larrañaga, C. Kuijpers, and R. Murga. Learning Bayesian network structures by searching for the best ordering with genetic algorithms. *IEEE Transactions on System, Man and Cybernetics*, 26:487–493, 1996.

[19] P. Larrañaga and Y. Yurramendi. Structure learning approaches in causal probabilistic networks, 1998.

[20] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, 50(2):157–224, 1988.

[21] Netica. Netica Bayesian network software from Norsys. http://www.norsys.com.

[22] A. J. Novobilski. The random selection and manipulation of legally encoded Bayesian networks in genetic algorithms. In *IC-AI*, pages 438–443, 2003.

[23] G. Provan and M. Singh. Learning Bayesian networks using feature selection. 1995.

[24] R. Robinson. Counting labeled acyclic digraphs. *New Directions in the Theory of Graphs, In Frank Harary, editor . Academic Press, New York*, pages 239–273, 1973.

[25] P. Spirtes, C. Glymour, and R. Scheines. Causation, Prediction and Search. *Lecture Notes in Statistics, New York: Springer Verlag*, 81, 1993.

[26] M. Teyssier and D. Koller. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In *Proceedings of the 21th Annual Conference on Uncertainty in Artificial Intelligence*, pages 584–59, Arlington, Virginia, 2005. AUAI Press.

[27] S. van Dijk, D. Thierens, and L. C. van der Gaag. Building a GA from design principles for learning Bayesian networks. In *GECCO*, pages 886–897, 2003.

[28] D. Whitley. The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989. Morgan Kaufman.

[29] M. L. Wong, S. Y. Lee, and K. S. Leung. A hybrid data mining approach to discover Bayesian networks using evolutionary programming. In *GECCO '02: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 214–222, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[30] O. Woodberry, A. Nicholson, K. Korb, and C. Pollino. Parameterising Bayesian networks. In *Australian Conference on Artificial Intelligence*, pages 1101–1107, 2004.