Two Novel Ant Colony Optimization Approaches for Bayesian Network Structure Learning

Yanghui Wu, John McCall, and David Corne

Abstract— **Learning Bayesian networks from data is an N-P hard problem with important practical applications. Several researchers have designed algorithms to overcome the computational complexity of this task. Difficult challenges remain however in reducing computation time for structure learning in networks of medium to large size and in understanding problem-dependent aspects of performance. In this paper, we present two novel algorithms (ChainACO and K2ACO) that use Ant Colony Optimization (ACO). Both algorithms search through the space of orderings of data variables. The ChainACO approach uses chain structures to reduce computational complexity of evaluation but at the expense of ignoring the richer structure that is explored in the K2ACO approach. The novel algorithms presented here are ACO versions of previously published GA approaches. We are therefore able to compare ACO vs GA algorithms and Chain vs K2 evaluations. We present a series of experiments on three well-known benchmark problems. Our results show problem-specific trade-offs between solution quality and computational effort.**

I. Introduction

BAYESIAN Networks (BNs) are probabilistic graphical models that represent the dependencies among a set of random variables in a chosen domain. A BN has two components: a structure and a set of parameters. The structure is a directed acyclic graph (DAG) and the parameters specify a joint probability distribution over the random variables [1]. BN is an efficient tool for performing probabilistic inference and is frequently utilized for modeling domain knowledge in Decision Support Systems, such as fault diagnosis, risk analysis, decision analysis, and medical expert systems [2][3][4]. However, learning Bayesian network is, in general, an NP-hard problem. It is known that the number of possible structures grows super-exponentially with the number of nodes [5], and so evaluating all possible structures is infeasible in most practical domains, where the number of variables is typically large. The process of finding cheaper approaches for learning the structure of BNs from large datasets is an area that has gained importance in recent years and is now widespread. In recent years, some powerful algorithms for the automatic learning of BNs from data have emerged. They can be grouped into two main approaches: methods based on conditional independence tests, and methods based on search and scoring metrics.

The conditional independence approach attempts to estimate from the data whether certain conditional dependences exist between the variables. These dependency relationships are usually measured through using statistical or information theoretical tests. Measures such as Pearson's product moment correlation coefficient, the Chi-Square test and mutual information are often used [6] [7] [8]. The search and score approach attempts to search for the best structure in the space of possible structures, according to a scoring function measuring the fit of each structure to the data. The structure found to have the best score is then returned [9]. Algorithms of this type may also require node ordering, in which a parent node precedes a child node so as to narrow the search space [9]. The most common scoring functions used in these algorithms include the K2-CH metric [9], BDeu [10], BIC [11], and Minimum Description Length (MDL) [12]. A range of well-known heuristic search techniques have been applied in this way, including: Hill Climbing [13], Genetic Algorithms [14], Simulated Annealing [15], and Ant Colony Optimization (ACO) [16] [17].

In this paper we are interested in the search and score approach and will investigate two novel algorithms to construct BN using Ant Colony Optimization. Previous research has already applied ACO to BN learning in a variety of ways [17] [18] [19] [20] [21]. The experimental results in these papers show better performance of ACO with complicated networks compared to other algorithms when reconstructing the test networks. ACO has been successfully applied to the Travelling Salesman Problem (TSP) [22]. The problem of finding an optimal ordering of variables in search and scoring approaches resembles the asymmetric TSP [23], [24], whose search space is the set of permutations of the nodes in a graph. In this paper, ACO will be used to search for the best node ordering on the domain that restricts the heuristic stage for constructing structure. We aim to empirically investigate to what extent ACO is an efficient way of searching node orderings.

The remainder of the paper is organized as follows. We briefly introduce the ACO for BN structure learning and K2

Manuscript received February 2010.

Y. Wu is with the IDEAS Research Institute, Robert Gordon University, Aberdeen, UK (e-mail: y.wu3@rgu.ac.uk).

J. McCall is with the IDEAS Research Institute, Robert Gordon University, Aberdeen, UK (Corresponding author: telephone +44(0)1224 262780 e-mail: jm@comp.rgu.ac.uk).

D. Corne is with the School of Mathematics and Computer Science, Heriot-Watt University, Edinburgh, UK (e-mail: dwcorne@macs.hw.ac.uk).

algorithm in section 2. In section 3, we describe our novel approaches in terms of the ACO metaheuristic. We then describe the experiments carried out for testing our approaches in section 4. In section 5 we discuss the results and finally we draw conclusions in section 6.

II. BACKGROUND

A. ACO for Bayesian Network Structure Learning

ACO [22] [25] [26] is a relatively new paradigm of bio-inspired algorithms. The inspiration for ACO is the cooperative behavior of real ants by means of chemical pheromone trails, which are able to find the shortest path from a food source to their nest. Artificial ants move through a construction space: a path in such a space defines a series of construction steps that can be use to assemble a solution to some problem of interest. ACO algorithm has shown very good performance when applied to solving such hard combinatorial problems as TSP, the quadratic assignment problem [27], routing problems [28], and sorting problems [29].

To date, there has been only a limited amount of research that applys ACO algorithms to learning BN. Examples of these approaches include the ACO-B[16], MMACO[18][19], and ACO-E[20][21].etc. These approaches typically integrate with other greedy construction heuristic algorithms, such as Algorithm B [10], K2SN algorithm [30] and the local discovery algorithm MMPC [31]. The algorithms that have been developed so far for BN structure learning have investigated appropriate representations of the problem, specific heuristic information to help guide the ants, ways of updating the pheromone track, and the probabilistic transition rule that move the ants to the next stage

We now review these existing ACO-based algorithms, emphasizing as we go those aspects particularly important to the approaches developed in this paper. These include: search space, nodes transitions rule, heuristic information, local and global updating rules. These ACO-based algorithms search in a range of different spaces. The space of orderings of the DAGs is searched in the ACO-K2SN algorithm; the graph made of DAGs with n nodes in ACO-B; the space of Equivalence Classes in the ACO-E algorithm; and the search space is the set of possible edges in the MMACO algorithm. All of these algorithms execute pheromone updating in two similar ways to that used on the TSP. One way is to updated step-by-step after each ant chooses a new node to visit, by reducing the amount of pheromone on the edge. The other way is global update at the end of each iteration, using the best solution found so far.

Each ACO-based algorithm has respective heuristic information η_{ij}, because the heuristic for the arc from node i to j often either relates to the scoring metric being used or relates to the search space. For example, in ACO-K2SN, where the space is the ordering of DAGs, the heuristic information is defined as

$$\eta_{ij} = \frac{1}{\left| f(x_j, P_a(x_j)) \right|}$$

where f is the scoring metric being used and $P_a(x_j)$, the parent of x_j are found by K2 algorithm. However, in MMACO algorithm, where the space is the set of possible edges, the heuristic information is defined as

$$\eta_{ij} = 1 + f\left(s_k \cup \langle E_{ij}, e \rangle\right) - f\left(s_k \cup \langle E_{ji}, e \rangle\right)$$

Here f is the scoring metric found by BDeu score function [10]. In choosing which node to visit as the next node for an ant lying at a particular node in a particular state, all these algorithms utilize the same probabilistic transition rule. The rule uses the values given by heuristic information and pheromone to decide which node to visit next. This is a random choice based on a distribution given by the heuristic information and pheromone distribution.

B. K2 Algorithm

In the search and score approach for learning BN structure, K2 is one of the best known algorithms that creates and evaluates a BN from a database of cases once an ordering between the system variables is given. For the evaluation of the joint probability of a network that it constructs, the formula of Cooper and Herskovits, the (K2-CH) metric, is used [9]. This metric can be expressed as

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

Here q_i denotes the number of possible different instances 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 P_{a_i} has its j th value. N_{ij} is the sum of all N_{ijk} for all values x_i can take. As it is easier to work in the logarithmic space, in practice, the scoring function uses the value $\log(P(B_S, D))$ instead of $P(B_S, D)$.

The K2-CH metric (1), decomposes into a product of factors:

$$F = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)} \prod_{k=1}^{r_i} N_{ijk}! \qquad (2)$$

Each factor expresses the possibility when x_i is the parent of x_j. Such factor evaluations require counting of value combinations throughout the dataset and so represent the main computational effort of the algorithm. We therefore

compare algorithm performance in terms of the number of such factor evaluations used to produce solutions of threshold quality.

III. CHAINACO AND K2ACO ALGORITHMS

We propose two novel algorithms using ACO for BNs structure learning based on two existing approaches ChainGA and K2GA. In this paper, we name the new approaches as ChainACO and K2ACO respectively.

ChainGA[32] is a search and score heuristic. It based on the hypothesis that a chain is a sufficiently good model to locate node orderings on which good BN structure can be built. The node ordering can be evaluated by building its associated chain structure and evaluating it given the data with scoring metric. This is a relatively cheap evaluation in terms of the number of K2-CH factor evaluations needed. The ChainGA for BN structure learning has two phases: in the first phase, a Genetic Algorithm (GA) is used to evolve the best ordering based on, chain model evaluations. Wherein the second, post-evolution phase, K2 is run sequentially on the best orderings found in the first phase and the best overall structure is returned. Previous results have shown that the ChainGA can get a significant reduction in computational cost.

The main idea of ChainACO approach comes from ChainGA. ChainACO also has two phases, depicted in Fig.1. In the first phase of ChainACO, we construct chains using ACO instead of GA. The second phase applies K2 to the best orderings found and returns the best structure. This is identical to the second phase in ChainGA.

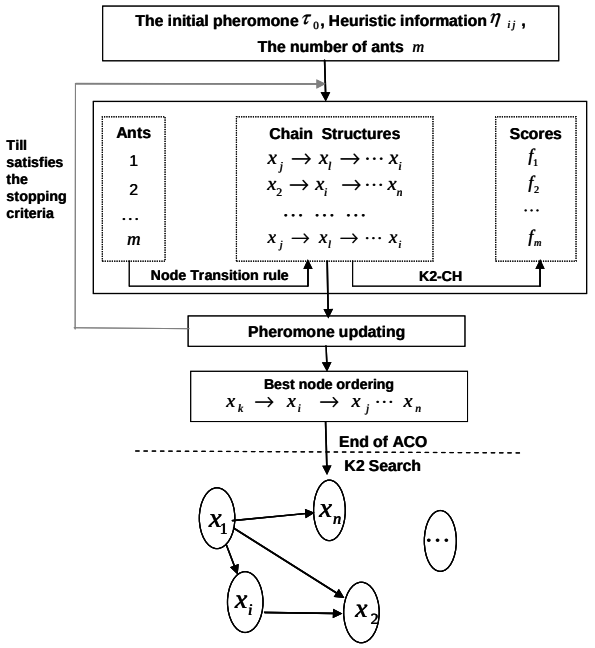

Fig. 1: ChainAC

The K2GA algorithm [33] similarly uses a GA to search the space of node orderings rather than the full space of BN structures. Here however there is only a single GA phase. The fitness of each ordering is calculated by running the K2 search algorithm on each ordering evaluated and returning the score of the network structure found. The best structure returned is that generated by K2 from the best ordering evaluated in this fashion.

Here we use ACO to replace the GA search in K2GA. In K2ACO, the initial individuals in the population are randomly created node orderings which are then optimized by a colony of ants in this space until a good ordering is found. During the ACO process, the fitness of each ordering is calculated by running the K2 search algorithm on it. Once the optimization terminates, as with K2GA, the structure corresponding to best ordering found is automatically obtained. Fig. 2 shows the operation of K2ACO.

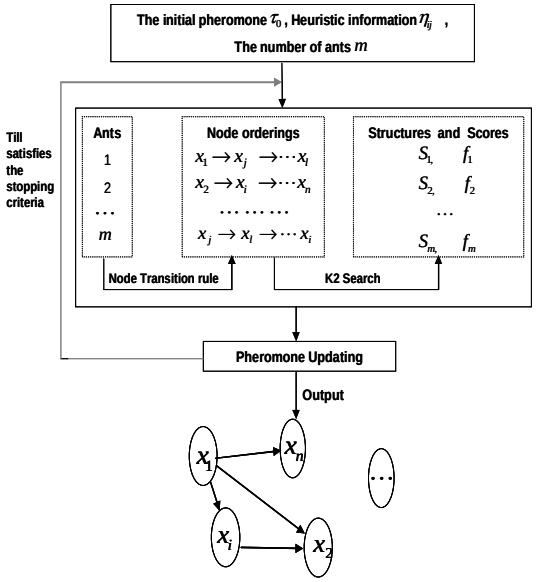

Fig. 2: K2ACO

The main components in ChainACO and K2ACO algorithms are defined as follows:

1) The search space can be regarded as a graph with n nodes, one for each random variable. The possible search space is the set of $n!$ orderings.

2) Node transitions rules. Ants in node i moves to node j according to the probabilistic state transition rule,

$$s = \begin{cases} \arg\max_{u \in J_k(i)} \left\{ [\tau(i,j)][\eta(i,j)]^{\beta} \right\} & \text{if } q \le q_0 \\ S & \text{otherwise} \end{cases} \quad (3)$$

Where $\tau(i, j)$ contains the pheromone amount deposited

in the arc which connect node i with node j. $\eta(i,j)$ represents heuristic information, which is a weighting function that assigns at each construction step a heuristic value to each feasible solution between any two nodes. q is a random number uniformly distributed in $[0,1]$, $q_0\,(0 \le q_0 \le 1)$ is a parameter, β is a parameter which control the relative importance of pheromone verse the probability, and S is random variable selected according to the probability given in (4).

$$p_k(i,j)=\begin{cases}\dfrac{[\tau(i,j)][\eta(i,j)]^{\beta}}{\sum_{u\in J_k(i)}\tau(i,j)[\eta(i,j)]^{\beta}} & \text{if } s\in J_k(i)\\ 0 & \text{otherwise}\end{cases} \quad (4)$$

In our approaches, the heuristic information η_{ij} in (3) and (4) is defined according to the possibility $g(i,j)$, which expresses the possibility when x_i is the parent of x_j, this is calculated from the K2-CH metric in our algorithm as defined in formula (2) .

$$g(i,j)=\prod_{j=1}^{q_i}\frac{(r_i-1)!}{(N_{ij}+r_i-1)}\prod_{k=1}^{r_i}N_{ijk}!$$

3) Local and updating rules. Ants prefer to move to nodes which are connected by high possibility $g(i,j)$ and a high amount of pheromone. Every time that an edge is chosen by one ant, the amount of pheromone is changed by applying the local updating rule (5):

$$\tau(i,j)\leftarrow(1-\rho)\tau(i,j)+\rho\cdot\tau_0 \quad (5)$$

The local updating rule is intended to avoid a very strong edge being chosen by all the ants. Once all ants have completed their tours a globe pheromone updating rule (6) is applied:

$$\tau(i,j)\leftarrow(1-\alpha)\tau(i,j)+\alpha\cdot\Delta\sigma(i,j) \quad (6)$$

Where

$$\Delta\sigma(i,j)=\begin{cases}\dfrac{1}{C(S^+)} & if\ \{ij\}\in S^+\\ \tau_{ij} & if\ \{ij\}\notin S^+\end{cases}$$

In (5) and (6), both ρ and $\alpha\,(0<\rho,\alpha\le 1)$ are parameters which control the pheromone evaporation and $C(S^+)$ is the cost associated the best solution.

4) The initial level of pheromone $\tau(i,j)$ in probabilistic state transition and in pheromone updating rules is defined in formula (7)

$$\tau_0=\frac{1}{n\cdot\left|f\left(G_{GA-K2}:D\right)\right|} \quad (7)$$

where n is the number of variables and $f(G_{GA-K2}:D)$ is the fitness which we obtained from [32].

The specific parameter values used are provided in the following section describing our experiments.

IV. EXPERIMENTAL EVALUATION

In order to measure the performance of the two proposed approaches, we carried out a set of experiments for both ChainACO and K2ACO algorithms. We also ran comparative experiments on ChainGA and K2GA algorithms. The aim of the experiments is to compare performance of four approaches to modeling three benchmarks data sets. We are also interested in the differences between algorithms based on ACO (ACO algorithms) and GA (GA algorithms), algorithms based on Chain (Chain algorithms) and K2 (K2 algorithms) respectively. Three benchmark problems have been selected in our research: Asia, Car and Alarm. The Asia network is a simple network with 8 binary nodes and 8 edges. It is a diagnostic demonstrative Bayesian network [34]. The Car Diagnostic Network consists of 18 nodes and 17 edges. It can be applied to diagnose malfunctioning of self-propelling vehicles [35]. The Alarm network is a medical diagnostic system for intensive care patient monitoring consisting of 37 nodes and 46 edges [36]. All the data cases are sampled using the Netica tool [37]. In this paper, the dataset sizes for Asia, Car and Alarm are 5000, 10000 and 3000 cases respectively.

A. Experimental Design

As described in section 3, both ChainACO and ChainGA algorithm have two phases. K2ACO and K2GA algorithms, on the other hand, each have a single phase. We set phase transition parameters and stopping conditions for each algorithm as follows:

For ChainACO and ChainGA algorithms, we specify a phase-transition parameter T such that when the population best chain structure score exceeds T, the algorithm transits to the second phase.

For K2ACO and K2GA algorithms, we specify a stopping condition T_0 according to previous experimental results in [32]. We run the ACO and GA iteration under the condition of the fitness of Bayesian network structure less than T_0 respectively to construct the best structures and get the best fitness at the same time.

In all cases the scoring metric used to evaluate the search is the K2-CH metric. The parameters used in ChainACO and K2ACO are shown in Table 1. The parameter settings of ChainGA and K2GA are shown in Table 2. These parameters are the same, for the GA algorithms, as those

used in [32].

TABLE 1: The ACO Parameter Settings

ACO Settings	Asia	Car	Alarm
Number of ant	6	8	8
Number of iteration	30	30	30
q_0	0.8	0.9	0.95
β	2	2	3
ρ, α	0.1	0.1	0.1
Number of Run	30	30	30

TABLE 2: The GA Parameter Settings

GA Settings	Asia	Car	Alarm
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
Number of Run	30	30	30

B. Experimental Results

For each algorithm, we recorded the following run data:

1) The value of the K2-CH metric (log version) in eq. (1). We know that this score associated with the best structure, as the score is the logarithm of a probability, and so it is negative number. The closer to zero, the closer the probability is to 1. This means the better is the network.

2) The number of factor evaluation (F.E.). F.E. is utilized to evaluate the efficiency of the algorithms [32]. It is defined as being the count of times the term (2) is accessed when Formula (1) is used. In ChainACO algorithm, the F.E. comes from three parts: the evaluations needed to generate initial heuristic information η_{ij} for node transitions rules, the evaluations needed to calculate chain structure scores in phase 1, and the evaluations needed in the K2 search phase for obtaining the structure score. In ChainGA, the F.E. score is made up of two parts: the evaluations needed in the chain structure phase and those in the K2 search phase. In K2ACO, the number of F.E. includes those needed for the initial heuristic information η_{ij} for node transitions rules and those needed for the K2 score evaluations. In K2GA, the F.E score is calculated solely from the K2 score evaluations needed.

3) The number of arcs found.

The results of our experiments are displayed in Tables 3-5. We have carried out 30 runs of each algorithm for each benchmark problem considered. The best score, F.E and their corresponding standard deviations are averaged over 30 runs. Success rates (S.R.) are obtained through comparison of the score of the best structure in each run with the target score. In our experiments, to the target is the stopping condition T_0, for example, in Asia network, $T_0 = -11150.00$. The arcs which we achieved correspond to the best-ever score obtained by each algorithm.

TABLE 3: Experimental Results for Asia ($T_0 = -11150.00$)

	Avg. Best Score	F. E.	S.R.	Arcs
ChainACO	-11167.33±18.89	**128.0±4.54**	50%	**8**
K2ACO	-11146.22±1.85	1166.3±630.05	90%	**8**
ChainGA	-11153.95±12.16	1133.43±197.24	57%	**8**
K2GA	**-11145.99±1.22**	2468.73±15.13	**100%**	**8**

TABLE 4: Experimental Results for Car ($T_0 = -23175.00$)

	Avg. Best Score	F. E.	S.R.	Arcs
ChainACO	-23186.77±18.04	2083.1±263.17	30%	23
K2ACO	-23163.83±4.07	2707.83±1784.1	60%	**20**
ChainGA	-23270±126.75	**964.83±198.46**	25%	**20**
K2GA	**-23163.17±7.52**	1774.63±468.07	**100%**	**20**

TABLE 5: Experimental Results for Alarm ($T_0 = -30110.00$)

	Avg. Best Score	F. E.	S.R.	Arcs
ChainACO	**-29814.2±90.99**	2480.07±1252	**100%**	**54**
K2ACO	-30073.3±94.99	7225.07±3405.1	57%	68
ChainGA	-30443.6±220.96	**1235.93±205.85**	10%	68
K2GA	-30087.7±182.31	2596.97±497.15	53%	63

V. DISCUSSION

Tables 3, 4, and 5 display the results obtained for the Asia, Car and Alarm networks, respectively. Table 6 displays normalized structural differences from the original network for each algorithm on each benchmark. This includes arcs added (A), arcs deleted (D), and arcs inverted (I). "Normalized" means here that, on each problem, the absolute number of each structural difference measured is divided by the corresponding number for ACOChain. Thus, a normlaised value is smaller than 1.00 denotes the corresponding approach performs better than ChainACO, whereas a value greater than 1.00 indicates a weaker performance than ChainACO.

With the aim of comparing overall computational efficiency, Table 7 displays the success rates per 10000 F.E. for each algorithm. Fig. 3, 4, and 5 plot the best scores over all runs for each algorithm for the Asia, Car, and Alarm

networks respectively.

TABLE 6: Normalized structural differences within four algorithms to three benchmarks model.

		ChainACO	K2ACO	ChainGA	K2GA
Asia	A	1.00	1.00	1.00	1.00
	D	1.00	1.00	1.00	1.00
	I	1.00	0.25	0.25	0.25
Car	A	1.00	0.50	0.63	0.50
	D	1.00	0.50	1.00	0.50
	I	1.00	1.00	1.00	1.25
Alarm	A	1.00	1.25	1.29	1.18
	D	1.00	0.95	1.00	1.00
	I	1.00	1.67	0.67	1.00

TABLE 7: Success rates per 10,000 F.E.

	Asia	Car	Alarm
ChainACO	**39.06**	1.44	**4.03**
K2ACO	7.72	2.22	0.79
ChainGA	5.03	2.59	0.81
K2GA	4.05	**5.64**	2.04

We now make detailed observations on the results for each benchmark problem:

For the Asia dataset, we carried out a one way ANOVA test using the Bonferroni correction with the four algorithms to compare the averaged score. The results indicate that there is a significant difference between the ChainACO approach and the other three approaches, $F(3,116)=22.819, P<0.05$. K2GA obtained the best structure score and the highest success rate, however, the number of F.E. is the biggest, almost twenty times that required by ChainACO. With respect to the number of F.E., ChainACO is the cheapest one; however, the success rate is the lowest. In order to show the trade-offs between efficiency and success rate, the success rate per 10000 function evaluations is shown across all experiments in Table 7. We observe that ChainACO provides the best trade-off for the Asia networks. Table 3 shows that each algorithm finds the same number of arcs. Table 6 displays the differences only in the inverted arcs. Comparing the Chain algorithms and K2 algorithms, we can find that the former is cheaper in Factor evaluation; this result is in accord with the hypothesis that a chain model is a sufficiently good model to locate node orderings of which good BN structure can be built [32]. The Box plots of the best scores found for the four algorithms across all experiments are displayed in Fig. 3.

For the Car diagnosis problem, the comparative results shown in Table 4 are notably different from those obtained on the Asia network. The one way ANOVA Bonferroni test with the four algorithms to compare the averaged score show that there is a significant difference between the K2GA approach and the other three approaches, $F(3,116)=18.035, P<0.05$. The success rates in Table 4 also are observably different, compared with K2GA, ChainACO and ChainGA have very low success rates. Therefore we might need to re-start these algorithms three and four times respectively to get the same quality of result as K2GA. From Table 7, it can be seen that the K2GA performs better than the three others. ChainACO obtained the lowest rate, the value is only 1.44. The structural arcs found in Table 4 and Table 6 show that ChainACO has more added and deleted arc than the other three algorithms. The results for the Car problem are that the GA outperforms ACO in computational effort and structure quality. Box plots of the best scores over all runs for each algorithm are displayed in Fig. 4.

For the Alarm problem, the difference is apparent in the Best score, F.E, success rate, and the arc found between ChainACO and other three algorithms. ANOVA tests using the Bonferroni correction show there is a significant difference between the ChainACO approach and the other three approaches, $F(3,116)=80.726, P<0.05$. From Table 6, we can find that the structural errors between ChainACO and other three algorithms also have significant differences. Table 7 shows that ChainACO obtained the best trade-off. It is interesting to note that K2ACO and ChainGA have significant differences in factor evaluation and success rates but very similar ratios. Both of them are only one fifths of the ChainACO's result. The experimental results for Alarm data also show that the K2 algorithms need more time in factor evaluation than Chain algorithms, the number of F.E. in K2ACO is nearly three times that for ChainACO, and in K2GA is nearly twice to ChainGA. The best scores found for the four algorithms are displayed in Fig. 5.

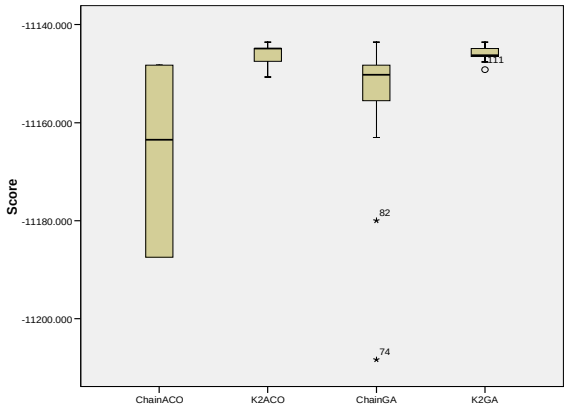

Fig. 3: The Box plots of the Best found Structure scores for Asia at each run for all algorithms

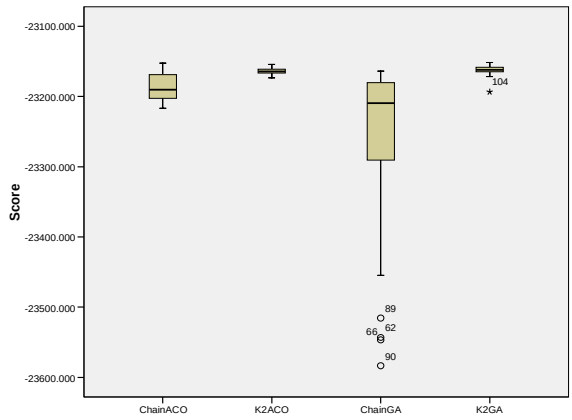

Fig. 4: The Box plots of the Best found Structure scores for Car at each run for all algorithms

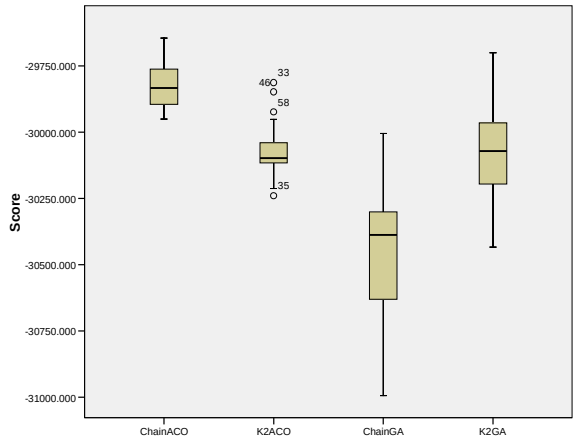

Fig. 5: The Box plots of the Best found Structure scores for Alarm at each run for all algorithms

VI. CONCLUSION

In this paper, we have proposed two new search and score algorithms for learning BN structures. Each utilizes the ACO metaheuristic to guide the search of node orderings. ChainACO uses ACO evolving on chain structures. This reduces computational expense of the search but with a penalty on the success in retrieving structure on known benchmarks K2ACO also searches node orderings using the K2 algorithm directly to construct and score locally optimized structures. K2ACO is more computationally expensive but in general has a higher success rate in recovering the known structures. We applied our approaches to three different benchmark datasets of varying complexities. We also compared our methods to two existing approaches which also search node orderings but utilizing genetic algorithms.

We conclude that there is a high degree of problem dependency both in the effectiveness and the efficiency of the approaches used. Both the choice of metaheuristic and the choice of scoring method can significantly affect performance. We also conclude that, for some problems, repeated restarts will be beneficial. This suggests that one way forward is to develop hyperheuristic approaches that adapt themselves to a particular problem as they learn. An important aspect of this will be to understand how particular structural features interact with particular heuristics to either promote or inhibit learning.

REFERENCES

[1] Heckerman, D., "A tutorial on learning with Bayesian networks in Learning in Graphical Models," pp.301-354, MIT Press, Cambridge, 1995

[2] Cano, R., Sordo, C., & Gutiérrez, J. M., "Applications of Bayesian networks in meteorology." Advances in Bayesian networks Springer-Verlag. pp. 309–327, 2004.

[3] Peter Lucas, "A model-based Approach to medicine Marking. K-P." Adlassnig (ed.), *in Proceedings of the EUNITE workshop on Intelligent Systems in patient Care*, Vienna, pp. 73-97.2001.

[4] Galán, S.F., F. Aguado, NasoNet, "Modeling the Spread of Nasopharyngeal Cancer with Networks of Probabilistic Events in Discrete Time," Artificial Intelligence in Medicine, vol. 25, 247-264. 2002.

[5] Robert W. Robinson Counting labeled acyclic digraphs. In F. Harary, editor, New Directions in the Theory of Graphs, Academic Press, New York. 239 -273,1973.

[6] J. Pearl. Causality: Models, Reasoning, and Inference. Cambridge University Press, 2000.

[7] Spirtes, P., Glymour, C., & Scheines, R., Causation, Prediction, and search. The MIT Press, second edition.2000.

[8] L.M. de Campos and J.F. Huete , A new approach for learning Bayesian networks using independence criteria. International Journal of Approximate Reasoning. vol. 24, pp. 11–37, 2000.

[9] Gregory F.Cooper,Edward Herskovits, A Bayesian Method for the Induction of Probabilistic Network from Data, Machine Learning,,v.9,no.4,pp.309-347,1992.

[10] Buntine, W. Theory refinement on Bayesian networks. UAI-17,Morgan Kaufmann,pp.52-60,1991.

[11] Schwartz, G. Estimating the dimensions of a model. Ann. Stat., 6, 461–464.1979.

[12] D. Heckerman, D. Geiger, and D. Chickering. "Learning Bayesian networks: The combination of knowledge and statistical data," *In KDD Workshop*, pp. 85--96, 1994

[13] Ioannis Tsamardinos, Laura E. BrownConstantin F. structure learning algorithm, Machine Learning, vol.6, no.1, pp. 31-78, 2006.

[14] Larrañaga P, Murga RH, Poza M, Kuijpers , "Structure learning of Bayesian networks by hybrid genetic algorithms," *In: Preliminary Papers of the Fifth International Workshop on Artificial Intelligence and Statistics*, pp. 310–316, 1995.

[15] T. Wang, J. Touchman, and G. Xue. "Applying two- level simulated annealing on Bayesian structure learning to infer genetic networks," In *Proceedings of the IEEE Computational Systems Bioinformatics Conference*, pp. 647– 648, 2004.

[16] L.M. de Campos, J.A. Gámez, and J.M. Puerta. Learning Bayesian network by ant colony optimisation: Searching in two different spaces. Mathware and Soft Computing IX (2-3), pp.251-268, 2002.

[17] L.M. de Campos, J.M. Fernández-Luna, J.A. Gámez, J.M. Puerta, Ant colony optimization for learning Bayesian network. International Journal of Approximate Reasoning, vol. 31, no.3, pp.291-311, 2002.

[18] Pedro C. Pinto, Andreas Nägele, Mathäus Dejori, Thomas A. Runkler, João Miguel da Costa Sousa: Learning of Bayesian networks by a local discovery ant colony algorithm. pp. 2741-2748, 2008.

[19] Pedro C. Pinto, Andreas Nagele, Mathaus Dejori, Thomas A. Runkler, and Joao M. C. Sousa, "Using a Local Discovery Ant Algorithm for Bayesian Network Structure Learning," *IEEE Transaction on Evolutionary Comptation*, VOL. 13, NO. 4, AUGUST 2009.

[20] Rónán Daly, Qiang Shen, and Stuart Aitken, Using ant colony optimization in learning Bayesian network equivalence classes. In Xue Z. Wang and Rui Fa Li, editors, *Proceedings of the 2006 UK Workshop on Computational Intelligence*, pp.111–118, 2006.

[21] Rónán Daly and Qiang Shen. Learning Bayesian network equivalence classes with ant colony optimization, Journal of Artificial Intelligence Research, vol. 35, pp.391–447. 2009.
[22] M. Dorigo and G. Di Caro, The ant colony optimization meta-heuristic. In: D. Corne, M. Dorigo and F. Glover, Editors, New Ideas in Optimization, McGraw-Hill, pp. 11–33.1999.
[23] C.J.Eyckelhof and M.Snoek. Ant systems for a dynamic TSP. In Marco Dorigo, Giani Di Caro, and Michael Sampels, editors, Ant Algorithms, LNCS, pp. 88- 99, 2002.
[24] Tsai CF, Tsai CW. "A new approach for solving large traveling salesman problem using evolution ant rules," *In: Neural Networks, IJCNN 2002, Proc. of the 2002 Int'l Joint Conf.* on, Vol 2. Honolulu: IEEE Press, 1540-1545, 2002.
[25] Marco Dorigoa, Christian Blum. Ant colony optimization theory: A survey. Theoretical Computer Science vol. 344, pp. 243 - 278. 2005.
[26] C. Blum, "Ant colony optimization: Introduction and recent trends," Phys. Life Reviews, vol. 2, pp. 353–373, 2005.
[27] M. Dorigo, L.M. Gambardella, M. Middendorf, T. Stutzle. Guest editorial: special section on ant colony optimization. IEEE Transactions on Evolutionary Computation 6:4, 317-319.2002.
[28] V. A. Cicirello and S. F. Smith, "Ant colony for autonomous decentralized shop floor routing," *in Proc. 5th Int. Symp. Autonomou Decentralized Syst*. pp. 383–390.2001.
[29] S. A. Hartmann and T. A. Runkler, "Online optimization of a color sorting assembly buffer using ant colony optimization," in Proc. Operations Res., pp. 415–420, 2007.
[30] L.M. de Campos and J.M. Puerta. Stochastic local search algorithms for learning belief networks: Searching in the space of orderings. Lecture Notes in Artificial Intelligence, 2143:228–239, 2001.
[31] I. Tsamardinos, L. F. Brown, and C. F. Aliferis, "The max-min hill climbing BN structure learning algorithm," Mach. Learning, vol. 65,no. 1, pp. 31–78, Oct. 2006.
[32] Ratiba Kabli, Frank Herrmann , John McCall, "A chain-model genetic algorithm for Bayesian network structure learning, Proceedings of the 9th annual conference on Genetic and evolutionary computation," *in IEEE World Congress on Computational Intelligence 2007*, London, England, July 07-11.
[33] 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: pp. 487--493, 1996.
[34] 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 vol.50, no.2, pp. 157--224, 1988.
[35] Bayesia. Bayesialab Bayesian network software. http://www.bayesia.com/.
[36] 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. pp. 247--256, 1989.
[37] Netica. Netica Bayesian network software from Norsys. http: //www.norsys.com.