Advanced Methodlogies for Bayesian Networks 1 (2017) 1-48 Submitted 4/00; Published 10/00

An Experimental Analysis of Anytime Algorithms for Bayesian Network Structure Learning

Colin Lee CDLEE@UWATERLOO.CA

Peter van Beek VANBEEK@UWATERLOO.CA

UNIVERSITY OF WATERLOO
WATERLOO (CANADA)

Abstract

Bayesian networks are a widely used graphical model with diverse applications in knowledge discovery, classification, and decision making. Learning a Bayesian network from discrete data can be cast as a combinatorial optimization problem and thus solved using optimization techniques—the well-known *score-and-search* approach. An important consideration when applying a score-and-search method for Bayesian network structure learning (BNSL) is its anytime behavior; i.e., how does the quality of the solution found improve as a function of the amount of time given to the algorithm. Previous studies of the anytime behavior of methods for BNSL are limited by the scale of the instances used in the evaluation and evaluate only algorithms that do not scale to larger instances. In this paper, we perform an extensive evaluation of the anytime behavior of the current state-of-the-art algorithms for BNSL. Our benchmark instances range from small (instances with fewer than 20 random variables) to massive (instances with more than 1,500 random variables). We find that a local search algorithm based on memetic search dominates the performance of other state-of-the-art algorithms when considering anytime behavior.

Keywords: Structure learning; score-and-search; anytime algorithms; memetic search.

1. Introduction

Bayesian networks are a popular probabilistic graphical model with diverse applications including knowledge discovery, prediction, and control. The structure of a Bayesian network (BN) can either be determined by a human domain expert or machine learned from discrete data. Bayesian network structure learning (BNSL) from discrete data is NP-hard in general to solve optimally but also NP-hard to solve approximately to within a reasonable factor (Chickering et al., 2003). Thus advanced search techniques are needed and the best methods for BNSL use a *score-and-search* approach where a scoring function is used to evaluate the quality of a proposed BN and the space of feasible solutions is systematically searched for a best-scoring BN.

Both global (exact) and local (approximate) search algorithms for BNSL have been studied extensively over the past two decades. Global search algorithms for BNSL include proposals based on dynamic programming (Koivisto and Sood, 2004; Silander and Myllymäki, 2006; Malone et al., 2011), integer linear programming (Jaakkola et al., 2010; Bartlett and Cussens, 2013), constraint programming (van Beek and Hoffmann, 2015), A* search (Yuan and Malone, 2013; Fan et al., 2014a; Fan and Yuan, 2015), depth-first branch-and-bound search (Tian, 2000; Malone and Yuan, 2014), and breadth-first branch-and-bound search (de Campos and Ji, 2011; Fan et al., 2014a,b; Fan and Yuan, 2015). Local search algorithms for BNSL include proposals based

on greedy search (Chickering et al., 1997), tabu search (Teyssier and Koller, 2005), ant colony optimization (De Campos et al., 2002), and memetic search (Lee and van Beek, 2017), over search spaces such as the space of network structures (Chickering et al., 1997), the space of equivalent network structures (Chickering, 2002), and the space of variable orderings (Teyssier and Koller, 2005; Scanagatta et al., 2015).

Our interest here is in anytime algorithms—algorithms where the quality of the solution improves over time. We perform an extensive evaluation of the behavior of the current state-of-the-art anytime algorithms for BNSL, and determine whether high quality solutions can be obtained at reasonable time cutoffs in the search. Our benchmark instances range from small (instances with fewer than 20 random variables) to massive (instances with more than 1,500 random variables). Our experimental study extends a recent study of anytime algorithms for BNSL by Malone and Yuan (2013) to include: (i) more varied and realistic benchmark instances, as their study was restricted to only synthetic benchmarks with 29 to 35 random variables using the BIC/MDL scoring function; (ii) a comparison to more current state-of-the-art global search algorithms; and (iii) a comparison to state-of-the-art local search algorithms, as their study omitted local search algorithms. We find that a local search algorithm based on memetic search dominates the performance of other state-of-the-art algorithms for BNSL when considering anytime behavior.

2. Background

In this section, we briefly review the necessary background in Bayesian networks before defining the Bayesian network structure learning problem (for more background on these topics see, for example, (Darwiche, 2009; Koller and Friedman, 2009)).

A Bayesian network (BN) is a probabilistic graphical model that consists of a labeled directed acyclic graph (DAG) in which the vertices $V = \{v_1, \ldots, v_n\}$ correspond to random variables, the edges represent direct influence of one random variable on another, and each vertex v_i is labeled with a conditional probability distribution $P(v_i \mid parents(v_i))$ that specifies the dependence of the variable v_i on its set of parents $parents(v_i)$ in the DAG. A BN can alternatively be viewed as a factorized representation of the joint probability distribution over the random variables and as an encoding of conditional independence assumptions.

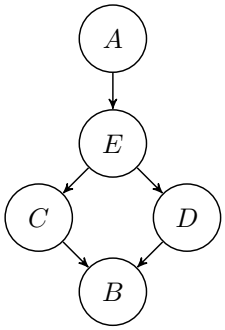
The predominant method for BN structure learning from data is the *score-and-search* method. Let G be a DAG over random variables V, and let $I = \{I_1, \ldots, I_N\}$ be a set of multivariate discrete data, where each instance I_i is an n-tuple that is a complete instantiation of the variables in V. A *scoring function* $\sigma(G \mid I)$ assigns a real value measuring the quality of G given the data I. Without loss of generality, we assume that a lower score represents a better quality network structure.

Definition 1 *Given a discrete data set $I = \{I_1, \ldots, I_N\}$ over random variables V and a scoring function σ, the* Bayesian network structure learning problem *is to find a directed acyclic graph G over V that minimizes the score $\sigma(G \mid I)$.*

Scoring functions balance goodness of fit to the data with a penalty term for model complexity to avoid overfitting. Common scoring functions include BIC/MDL (Schwarz, 1978; Lam and Bacchus, 1994) and BDeu (Buntine, 1991; Heckerman et al., 1995). An important property of these (and most) scoring functions is decomposability, where the score of the entire network $\sigma(G \mid I)$ can be rewritten as the sum of local scores $\sum_{i=1}^{n} \sigma(v_i, parents(v_i) \mid I)$ that only depend on v_i and the parent set of v_i in G. A common assumption is that the local score $\sigma(v_i, p \mid I)$ for each possible

$A:$	$\{D\}$, 9.6	$\{C\}$, 9.9	$\{E\}$, 10.0	$\{\}$, 15.4	
$B:$	$\{C, D\}$, 12.1	$\{C\}$, 12.2	$\{E\}$, 12.3	$\{\}$, 14.1	
$C:$	$\{E\}$, 3.6	$\{D\}$, 5.2	$\{A, B\}$, 10.9	$\{A\}$, 11.4	$\{\}$, 17.0
$D:$	$\{E\}$, 3.6	$\{C\}$, 5.2	$\{A, B\}$, 10.9	$\{A\}$, 11.4	$\{\}$, 17.0
$E:$	$\{D\}$, 3.7	$\{A\}$, 4.2	$\{A, B\}$, 11.2	$\{C\}$, 11.6	$\{\}$, 17.0

(a)



(b)

Figure 1: (a) Random variables and possible parent sets for Example 1; (b) minimum cost DAG structure with cost 38.9.

parent set $p \subseteq 2^{V-\{v_i\}}$ and each random variable v_i has been computed in a preprocessing step prior to the search for the best network structure. Pruning techniques can be used to reduce the number of possible parent sets that need to be considered, but in the worst-case the number of possible parent sets for each variable v_i is exponential in n, where n is the number of vertices in the DAG.

Example 1 *Let A, B, C, D, and E be random variables with the possible parent sets and associated scores shown in Figure 1(a). For example, if the parent set $\{C, D\}$ for random variable B is chosen there would be a directed edge from C to B and a directed edge from D to B and those would be the only incoming edges to B. The local score for this parent set is 12.1. If the parent set $\{\}$ for random variable A is chosen, there would be no incoming edges to A; i.e., A would be a source vertex. Figure 1(b) shows the minimum cost DAG with cost* $15.4 + 4.2 + 3.6 + 3.6 + 12.1 = 38.9$.

Note that, although the Bayesian network structure learning problem is NP-hard, once the best network structure or DAG has been chosen, it is an easy next step to estimate from complete data the conditional probability distributions that label each vertex.

3. Anytime Search Algorithms for BNSL

In this section, we briefly review the state-of-the-art for anytime search algorithms for BNSL.

Many of the current state-of-the-art global and local search algorithms for BNSL are based on the search space of all permutations first proposed by Larrañaga et al. (1996). The search space consists of all possible permutations of the random variables and relies on the fact that for a fixed permutation of the random variables, finding the minimal cost network for that permutation is straightforward. The algorithms of course differ in how they traverse the space of all permutations and whether they guarantee exact solutions (global search algorithms) or only approximate solutions (local search algorithms).

For global search algorithms, the experimental evaluation in van Beek and Hoffmann (2015), shows that methods based on integer linear programming (Bartlett and Cussens, 2013), constraint programming (van Beek and Hoffmann, 2015), and A* search (Yuan and Malone, 2013; Fan et al., 2014a; Fan and Yuan, 2015) are in contention for the state-of-the-art. The latter two methods search the space of all permutations. Of these three methods, only A* is not an anytime algorithm as no

solution is reported until the algorithm terminates. Although, as shown by Malone and Yuan (2013), adaptations can add anytime behavior to A*, we do not pursue the A* approach further in our evaluation of anytime algorithms as experiments show that the A* method does not scale quite as well as the competing approaches, as it runs out of memory sooner on larger instances (in fairness, the scalability of the A* approach on a very large memory machine is still somewhat of an open question). Thus, for global search algorithms, in our experiments we evaluate GOBNILP, Bartlett and Cussens's implementation of their integer linear programming approach, and CPBayes, van Beek and Hoffmann's implementation of their constraint programming approach. The implementation of CPBayes was modified to use the improved upper bound provided by the local search algorithm MINOBS (see below).

For local search algorithms, the experimental evaluations in Scanagatta et al. (2015) and Lee and van Beek (2017), shows that algorithms based on a search space that consists of all permutations are the state-of-the-art. Teyssier and Koller (2005) apply the permutation space within a local search algorithm for BNSL and give a tabu search algorithm for BNSL that performs adjacent swaps in the permutation to define a search neighborhood. Building on the work of Teyssier and Koller (2005), Scanagatta et al. (2015) show how to significantly improve the search of the local neighborhood and Lee and van Beek (2017) show how to significantly improve the search by doing insertions rather than adjacent swaps and using a memetic or population-based approach. Thus, for local search algorithms, in our experiments we evaluate ASOBS, Scanagatta et al.'s implementation of their approach, and MINOBS, Lee and van Beek's implementation of their memetic approach.

4. Experimental Evaluation

In this section, we present our computational study. We begin by presenting the experimental setup, followed by the experimental results.

4.1 Experimental setup

The sets of benchmark instances used in our study were obtained as follows.

- *Instances reported in Tables 2–5*: These instances were computed from data sets obtained from J. Cussens, B. Malone, the UCI Machine Learning Repository[1], and data generated from networks obtained from the Bayesian Network Repository[2]. The local scores were computed from the data sets using code provided by B. Malone[3]. The BIC/MDL (Schwarz, 1978; Lam and Bacchus, 1994) and BDeu (Buntine, 1991; Heckerman et al., 1995) scoring methods were used on these data sets. For some of the larger BDeu instances, the maximum indegree of the parent sets was restricted to be 8, and for the largest BDeu instances, the maximum indegree was restricted to be 6 in order to complete the computation of the local scores for a data set within 24 hours of CPU time. For all other instances, the maximum indegree was unrestricted.
- *Instances reported in Tables 6–8*: These instances, and the accompanying discrete data, were obtained from the the Bayesian Network Learning and Inference Package (BLIP)[4]. The

1. http://archive.ics.uci.edu/ml/
2. http://www.bnlearn.com/bnrepository/
3. http://urlearning.org/
4. http://blip.idsia.ch/

Table 1: Notation used in Tables 2–8.

symbol	meaning
n	number of random variables in the data set
N	number of instances in the data set
d	total number of possible parents sets for the random variables
—	indicates method did not report any solution within the given time bound
opt	indicates method found the known optimal solution within the given time bound
benchmark*	indicates optimal value for benchmark is not known; in such cases the percentage from optimal is calculated using best value found within 24 hours of CPU time

BIC/MDL (Schwarz, 1978; Lam and Bacchus, 1994) scoring method was used and the maximum indegree of the parents sets was restricted to be 6.

As in previous work, the local score for each possible parent set for each random variable was computed in a preprocessing step from discrete data (either by us or by others) prior to the search for the best network structure and we do not report the preprocessing time. One justification for computing all of the possible parent sets in a preprocessing step, rather than on an as-needed basis during the search, is that the computation of the parent sets is embarrassingly parallel and one can easily take advantage of additional resources to speed the computation.

The anytime algorithms for BNSL evaluated in our study were the following:

- GOBNILP[5], version 1.6.2 (Bartlett and Cussens, 2013, 2017);
- CPBayes[6], version 1.2 (van Beek and Hoffmann, 2015);
- ASOBS[7], version of December 2016 (Scanagatta et al., 2015); and
- MINOBS[8], version 0.2 (Lee and van Beek, 2017).

GOBNILP, CPBayes, and MINOBS are all implemented in C/C++. ASOBS is written in Java and therefore runs more slowly compared to these other methods. However, the largest time bounds in the experiments involving ASOBS are long enough that the method appears to stagnate—improving solutions stop being found at approximately the half way point of the time bound. Therefore, it appears unlikely that these results would change in a significant way if the method were to be made faster. The experiments for all methods other than ASOBS were run on a single core of an AMD Opteron 275 @ 2.2 GHz. Each run was allotted a maximum 30 GB of memory. Due to software limited availability, tests for ASOBS were run on a restricted set of instances courtesy of M. Scanagatta with the same memory limits and on a single core of an AMD Opteron 2350 @ 2.0 GHz. These two processors have similar single core performance. The methods were run with their default values. For MINOBS, ten tests with different random seeds were tested for each instance and the median is reported. GOBNILP, CPBayes, and ASOBS were only run once due to time constraints.

5. https://www.cs.york.ac.uk/aig/sw/gobnilp/
6. https://cs.uwaterloo.ca/~vanbeek/Research
7. http://blip.idsia.ch/
8. https://github.com/kkourin/mobs/releases/tag/0.2

Table 2: *BIC scoring function, small networks ($n \leq 20$ random variables).* Percentage from optimal on each benchmark, for various time bounds and solution methods: GOBNILP v1.6.2 (Bartlett and Cussens, 2013), CPBayes v1.1 (van Beek and Hoffmann, 2015), and MINOBS v0.2 (Lee and van Beek, 2017).

				1 minute			5 minutes			10 minutes		
benchmark	n	N	d	GO	CP	MI	GO	CP	MI	GO	CP	MI
nltcs	16	3,236	7,933	0.2%	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
msnbc	17	58,265	47,229	—	*opt*	*opt*	0.4%	*opt*	*opt*	0.0%	*opt*	*opt*
letter	17	20,000	4,443	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
voting	17	435	1,848	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
zoo	17	101	554	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
tumour	18	339	219	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
lympho	19	148	143	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
vehicle	19	846	763	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
hepatitis	20	155	266	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
segment	20	2,310	1,053	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*

Table 3: *BDeu scoring function, small networks ($n \leq 20$ random variables).* Percentage from optimal on each benchmark, for various time bounds and solution methods: GOBNILP v1.6.2 (Bartlett and Cussens, 2013), CPBayes v1.1 (van Beek and Hoffmann, 2015), and MINOBS v0.2 (Lee and van Beek, 2017).

				1 minute			5 minutes			10 minutes		
benchmark	n	N	d	GO	CP	MI	GO	CP	MI	GO	CP	MI
nltcs	16	3,236	8,091	0.0%	*opt*	*opt*	0.0%	*opt*	*opt*	*opt*	*opt*	*opt*
msnbc	17	58,265	50,921	—	*opt*	*opt*	0.2%	*opt*	*opt*	0.1%	*opt*	*opt*
letter	17	20,000	18,841	1.3%	*opt*	*opt*	0.1%	*opt*	*opt*	0.0%	*opt*	*opt*
voting	17	435	1,940	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
zoo	17	101	2,855	1.7%	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
tumour	18	339	274	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
lympho	19	148	345	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
vehicle	19	846	3,121	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
hepatitis	20	155	501	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
segment	20	2,310	6,491	0.3%	*opt*	*opt*	0.3%	*opt*	*opt*	0.0%	*opt*	*opt*

4.2 Experimental results

The algorithms are compared on the quality of the solution reported at various time bounds. The quality of the solution was measured by either the percentage from optimal, if known, and otherwise the percentage from the best solution found within 24 hours of CPU time by any method. The Bayesian Network Repository classifies networks as small ($n \leq 20$ variables), medium ($20 < n \leq 60$ variables), large ($60 < n \leq 100$ variables), very large ($100 < n \leq 1000$ variables), and massive ($n > 1000$ variables). We categorize our experimental results by the size of the networks and by the scoring function. Table 1 shows the notation used in reporting our results.

Table 4: *BIC scoring function, medium networks (*$20 < n \leq 60$ *random variables).* Percentage from optimal on each benchmark, for various time bounds and solution methods: GOBNILP v1.6.2 (Bartlett and Cussens, 2013), CPBayes v1.1 (van Beek and Hoffmann, 2015), and MINOBS v0.2 (Lee and van Beek, 2017).

				1 minute			5 minutes			10 minutes		
benchmark	n	N	d	GO	CP	MI	GO	CP	MI	GO	CP	MI
mushroom	23	8,124	13,025	1.1%	*opt*	*opt*	0.6%	*opt*	*opt*	0.6%	*opt*	*opt*
autos	26	159	2,391	1.5%	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
insurance	27	1,000	506	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
horse colic	28	300	490	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
steel	28	1,941	93,026	—	0.0%	0.0%	0.9%	*opt*	*opt*	0.7%	*opt*	*opt*
flag	29	194	741	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
wdbc	31	569	14,613	0.7%	*opt*	*opt*	0.2%	*opt*	*opt*	0.2%	*opt*	*opt*
water	32	1,000	159	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
mildew	35	1,000	126	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
soybean	36	266	5,926	1.6%	*opt*	*opt*	1.6%	*opt*	*opt*	*opt*	*opt*	*opt*
alarm	37	1,000	1,002	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
bands	39	277	892	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
spectf	45	267	610	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
sponge	45	76	618	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
barley	48	1,000	244	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
hailfinder	56	100	50	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
hailfinder	56	500	43	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
lung cancer	57	32	292	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
carpo	60	100	423	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
carpo	60	500	847	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*

Tables 2 & 3 show the results for small networks ($n \leq 20$ random variables) for the BIC and BDeu scoring functions, respectively, and the GOBNILP, CPBayes, and MINOBS methods. CPBayes and MINOBS are able to consistently find optimal solutions within a 1 minute time bound, whereas GOBNILP sometimes is unable to find its first solution within 1 minute and sometimes has not yet found the optimal solution with a 10 minute time bound. Of course, GOBNILP and CPBayes, being global search methods, may terminate earlier than a time limit once a solution has been proven optimal, whereas ASOBS and MINOBS, being local search methods, terminate only when a time bound is reached. The parameter d, the total number of possible parents sets across all the random variables, is a relatively good predictor for the instances that GOBNILP finds difficult among these small networks as it strongly correlates with the size of the integer programming model.

Tables 4 & 5 show the results for medium networks ($20 < n \leq 60$ random variables) for the BIC and BDeu scoring functions, respectively, and the GOBNILP, CPBayes, and MINOBS methods. Note the difference in time bounds between instances using BIC scoring and the instances using BDeu scoring; BDeu scoring leads to instances that are significantly harder to solve. In fact, for two of the BDeu instances, soybean and lung cancer, neither of the exact methods could determine the optimal solution. As well, the BDeu instances themselves are hard to compute (i.e., the preprocessing step that computes the local score for each possible parent set for each random

Table 5: *BDeu scoring function, medium networks (*$20 < n \leq 60$ *random variables).* Percentage from optimal on each benchmark, for various time bounds and solution methods: GOBNILP v1.6.2 (Bartlett and Cussens, 2013), CPBayes v1.1 (van Beek and Hoffmann, 2015), and MINOBS v0.2 (Lee and van Beek, 2017).

				5 minutes			1 hour			12 hours		
benchmark	n	N	d	GO	CP	MI	GO	CP	MI	GO	CP	MI
mushroom	23	8,124	438,185	—	0.0%	0.0%	0.5%	*opt*	0.0%	0.1%	*opt*	*opt*
autos	26	159	25,238	4.3%	0.0%	0.0%	1.2%	*opt*	0.0%	*opt*	*opt*	*opt*
insurance	27	1,000	792	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
horse colic	28	300	490	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
steel	28	1,941	113,118	2.0%	0.0%	*opt*	0.5%	*opt*	*opt*	0.4%	*opt*	*opt*
flag	29	194	1,324	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
wdbc	31	569	13,473	0.6%	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
water	32	1,000	261	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
mildew	35	1,000	166	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
soybean*	36	266	212,425	—	0.1%	0.1%	3.1%	0.1%	0.1%	1.8%	0.0%	0.0%
alarm	37	1,000	2,113	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
bands	39	277	1,165	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
spectf	45	267	316	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
sponge	45	76	10,790	0.4%	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
barley	48	1,000	364	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
hailfinder	56	100	199	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
hailfinder	56	500	447	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*
lung cancer*	57	32	22,338	6.7%	0.3%	0.1%	6.7%	0.0%	0.0%	0.9%	0.0%	0.0%
carpo	60	100	15,408	2.1%	*opt*	*opt*	0.5%	*opt*	*opt*	*opt*	*opt*	*opt*
carpo	60	500	3,324	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*	*opt*

variable from discrete data prior to search), where the larger instances required 24 hours of CPU time to compute the parent sets even under restrictions on the indegree of the parent sets.

CPBayes and MINOBS are able to consistently find optimal or near-optimal solutions within a 1 minute time bound for the BIC instances and a 5 minute time bound for the BDeu instances. After 5 minutes and 1 hour, respectively, the solutions found by these methods are almost all optimal. By the largest time bound, 10 minutes and 12 hours, respectively, for all of the instances where the optimal solution was known, CPBayes and MINOBS found the optimal solution, whereas for five of these instances GOBNILP found high-quality solutions but was unable to find the optimal solution. Once again, the parameter d is a good predictor for the instances that GOBNILP finds difficult. (It should be noted, however, that GOBNILP is able to prove the optimality of larger instances than CPBayes, and thus GOBNILP scales better on the parameter n.) In summary though, all three methods are competitive on the medium networks ($20 < n \leq 60$ random variables) in terms of anytime behavior. However, as will be seen next, the medium networks are near the limits of exact solvers such as GOBNILP and CPBayes.

Tables 6, 7 & 8 show the results for large ($60 < n \leq 100$ variables), very large ($100 < n \leq 1000$ variables), and massive networks ($n > 1000$ variables), and the GOBNILP, CPBayes, ASOBS, and MINOBS methods. Results are reported for only the BIC scoring function, as instances for the BDeu scoring function could not be computed within a 24 hour limit on CPU time.

Table 6: *BIC scoring function, large networks (*$60 < n \leq 100$ *random variables).* Percentage from optimal on each benchmark, for various time bounds and solution methods: GOBNILP v1.6.2 (Bartlett and Cussens, 2013), CPBayes v1.1 (van Beek and Hoffmann, 2015), ASOBS (Scanagatta et al., 2015), and MINOBS v0.2 (Lee and van Beek, 2017).

				1 hour				12 hours			
benchmark	n	N	d	GO	CP	AS	MI	GO	CP	AS	MI
kdd	64	34,955	152,873	3.4%	*opt*	0.5%	0.0%	3.3%	*opt*	0.5%	*opt*
plants*	69	3,482	520,148	44.5%	0.1%	17.5%	0.0%	33.0%	0.0%	14.8%	0.0%
bnetflix	100	3,000	1,103,968	—	*opt*	3.7%	*opt*	—	*opt*	2.2%	*opt*

Table 7: *BIC scoring function, very large networks (*$100 < n \leq 1000$ *variables).* Percentage from optimal on each benchmark, for various time bounds and solution methods: GOBNILP v1.6.2 (Bartlett and Cussens, 2013), CPBayes v1.1 (van Beek and Hoffmann, 2015), ASOBS (Scanagatta et al., 2015), and MINOBS v0.2 (Lee and van Beek, 2017).

				1 hour				12 hours			
benchmark	n	N	d	GO	CP	AS	MI	GO	CP	AS	MI
accidents*	111	2,551	1,425,966	—	0.6%	325.6%	0.3%	—	0.0%	155.9%	0.0%
pumsb_star*	163	2,452	1,034,955	320.7%	—	24.0%	0.0%	277.2%	—	18.9%	0.0%
dna*	180	1,186	2,019,003	—	—	7.3%	0.4%	—	—	5.8%	0.0%
kosarek*	190	6,675	1,192,386	—	—	8.4%	0.1%	—	—	8.0%	0.0%
msweb*	294	5,000	1,597,487	—	—	1.5%	0.0%	—	—	1.3%	0.0%
diabetes*	413	5,000	754,563	—	—	0.8%	0.0%	—	—	0.7%	0.0%
pigs*	441	5,000	1,984,359	—	—	16.8%	1.8%	—	—	16.8%	0.1%
book*	500	1,739	2,794,588	—	—	9.9%	0.8%	—	—	9.1%	0.1%
tmovie*	500	591	2,778,556	—	—	36.1%	5.5%	—	—	33.4%	0.2%
link*	724	5,000	3,203,086	—	—	28.4%	0.2%	—	—	17.1%	0.1%
cwebkb*	839	838	3,409,747	—	—	32.4%	2.3%	—	—	25.5%	0.2%
cr52*	889	1,540	3,357,042	—	—	25.9%	2.2%	—	—	23.5%	0.1%
c20ng*	910	3,764	3,046,445	—	—	16.3%	1.0%	—	—	14.6%	0.0%

Table 8: *BIC scoring function, massive networks (*$n > 1000$ *random variables).* Percentage from optimal on each benchmark, for various time bounds and solution methods. GOBNILP v1.6.2 (Bartlett and Cussens, 2013), CPBayes v1.1 (van Beek and Hoffmann, 2015), ASOBS (Scanagatta et al., 2015), and MINOBS v0.2 (Lee and van Beek, 2017).

				1 hour				12 hours			
benchmark	n	N	d	GO	CP	AS	MI	GO	CP	AS	MI
bbc*	1,058	326	3,915,071	—	—	26.0%	4.5%	—	—	24.4%	0.5%
ad*	1,556	487	6,791,926	—	—	15.2%	3.2%	—	—	15.0%	0.5%

The global search solvers GOBNILP and CPBayes are not competitive one these large to massive networks. GOBNILP is able to find solutions for only three of the instances, and for the other instances the memory requirements exceed the limit of 30 GB. CPBayes is able to find solutions

for only four of the instances and this is only due to the high-quality initial upper bound found by MINOBS (recall that we modified the implementation of CPBayes to use the improved upper bound provided by MINOBS, rather than its existing simple hill climbing approach). As well, it should be noted that CPBayes can only handle instances for $n \leq 128$ as this is a fundamental limitation of how it represents parent sets and how it computes lower bounds.

The local search solvers ASOBS and MINOBS are able to scale to and find solutions for all of these large to massive instances within reasonable time bounds. The local search algorithm MINOBS performs exceptionally well, consistently finding high-quality solutions within 1 hour and very high-quality solutions within 12 hours. (Recall that for MINOBS, ten tests with different random seeds were tested for each instance. As an example to quantify the consistency of MINOBS, the standard deviation of each of the results using a 12 hour cutoff is bounded by 0.3.) Unfortunately, it is unknown whether there is still room for improvement as for only two of these instances is the optimal solution known. Notably, the local search algorithm ASOBS does not perform as well on these large instances, often reporting solutions that are quite far from optimal for each of the time bounds. As reported above, ASOBS appears to stagnate well before the 12 hour time bound and improving solutions stop being found at approximately the 6 hour time point. This suggests that longer time bounds would not significantly improve the quality of the solutions found by ASOBS.

5. Conclusion

An important consideration when applying a score-and-search method for Bayesian network structure learning (BNSL) is its anytime behavior; i.e., how does the quality of the solution found improve as a function of the amount of time given to the algorithm. We performed an extensive evaluation of the anytime behavior of the current state-of-the-art algorithms for BNSL. Our benchmark instances range from small (instances with fewer than 20 random variables) to massive (instances with more than 1,500 random variables). We find that MINOBS, a local search algorithm based on memetic search, dominates the performance of other state-of-the-art algorithms when considering anytime behavior. On small instances MINOBS quickly reports optimal solutions; on medium instances MINOBS finds optimal or near-optimal solutions; and on large, very large, and massive instances MINOBS finds solutions that are of significantly higher quality than the competing state-of-the-art algorithms.

References

Mark Bartlett and James Cussens. Advances in Bayesian network learning using integer programming. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence*, pages 182–191, 2013.

Mark Bartlett and James Cussens. Integer linear programming for the Bayesian network structure learning problem. *Artificial Intelligence*, 244:258–271, 2017.

Wray L. Buntine. Theory refinement of Bayesian networks. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 52–60, 1991.

David M. Chickering. Learning equivalence classes of Bayesian network structures. *J. Mach. Learn. Res.*, 2:445–498, 2002.

David M. Chickering, David Heckerman, and Christopher Meek. A Bayesian approach to learning Bayesian networks with local structure. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 80–89, 1997.

David M. Chickering, Christopher Meek, and David Heckerman. Large-sample learning of Bayesian networks is NP-hard. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, pages 124–133, 2003.

Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

Cassio P. de Campos and Qiang Ji. Efficient structure learning of Bayesian networks using constraints. *J. Mach. Learn. Res.*, 12:663–689, 2011.

Luis M. De Campos, Juan M. Fernández-Luna, José A. Gámez, and José M. Puerta. Ant colony optimization for learning Bayesian networks. *International J. of Approximate Reasoning*, 31: 291–311, 2002.

Xiannian Fan and Changhe Yuan. An improved lower bound for Bayesian network structure learning. In *Proceedings of the 29th Conference on Artificial Intelligence*, 2015.

Xiannian Fan, Brandon Malone, and Changhe Yuan. Finding optimal Bayesian network structures with constraints learned from data. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, pages 200–209, 2014a.

Xiannian Fan, Changhe Yuan, and Brandon Malone. Tightening bounds for Bayesian network structure learning. In *Proceedings of the 28th Conference on Artificial Intelligence*, pages 2439–2445, 2014b.

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

Tommi Jaakkola, David Sontag, Amir Globerson, and Marina Meila. Learning Bayesian network structure using LP relaxations. In *Proceedings of the International Conf. on Artificial Intelligence and Statistics (AISTATS-10)*, pages 358–365, 2010.

Mikko Koivisto and Kismat Sood. Exact Bayesian structure discovery in Bayesian networks. *J. Mach. Learn. Res.*, 5:549–573, 2004.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

Wai Lam and Fahiem Bacchus. Using new data to refine a Bayesian network. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 383–390, 1994.

Pedro Larrañaga, Cindy M. H. Kuijpers, Roberto H. Murga, and Yosu Yurramendi. Learning Bayesian network structures by searching for the best ordering with genetic algorithms. *IEEE Transactions on System, Man and Cybernetics*, 26:487–493, 1996.

Colin Lee and Peter van Beek. Metaheuristics for score-and-search Bayesian network structure learning. In *Proceedings of the 30th Canadian Conference on Artificial Intelligence*, pages 129–141, 2017. Available as: LNCS 10233.

Brandon Malone and Changhe Yuan. Evaluating anytime algorithms for learning optimal Bayesian networks. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence*, pages 381–390, 2013.

Brandon Malone and Changhe Yuan. A depth-first branch and bound algorithm for learning optimal Bayesian networks. In *Graph Structures for Knowledge Representation and Reasoning*, volume 8323 of *Lecture Notes in Computer Science*, pages 111–122. Springer, 2014.

Brandon Malone, Changhe Yuan, and Eric A. Hansen. Memory-efficient dynamic programming for learning optimal Bayesian networks. In *Proceedings of the 25th Conference on Artificial Intelligence*, pages 1057–1062, 2011.

Mauro Scanagatta, Cassio P. de Campos, Giorgio Corani, and Marco Zaffalon. Learning Bayesian networks with thousands of variables. In *Advances in Neural Information Processing Systems*, pages 1864–1872, 2015.

Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6:461–464, 1978.

Tomi Silander and Petri Myllymäki. A simple approach for finding the globally optimal Bayesian network structure. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, pages 445–452, 2006.

Marc Teyssier and Daphne Koller. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pages 548–549, 2005.

Jin Tian. A branch-and-bound algorithm for MDL learning Bayesian networks. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 580–588, 2000.

Peter van Beek and Hella-Franziska Hoffmann. Machine learning of Bayesian networks using constraint programming. In *Proceedings of the 21st International Conference on Principles and Practice of Constraint Programming*, pages 428–444, 2015.

Changhe Yuan and Brandon Malone. Learning optimal Bayesian networks: A shortest path perspective. *J. of Artifical Intelligence Research*, 48:23–65, 2013.