An Algorithm For Generation of Large Bayesian Networks

A. R. Statnikov, I. Tsamardinos, C.F. Aliferis

Department of Biomedical Informatics,
Discovery Systems Laboratory,
Vanderbilt University

Technical Report DSL-03-01
May 28th 2003

Abstract. We propose a novel algorithm and software for the generation of arbitrarily large Bayesian networks (e.g., graphical models representing joint probability distributions) by tiling smaller real-world known networks (tiles). The algorithm adds edges between tiles, while preserving joint probability distribution of the tiles. This technique will allow researchers to conduct large-scale evaluation of learning algorithms on the real-world large Bayesian networks which was impossible due to unavailability of such networks.

1. Introduction

Bayesian networks (BNs), graphical models of joint probability distributions, are currently widely used in a variety of research and industrial areas. A major branch of BN research is learning networks from the data [6]. Although BN learning is an NP-hard problem [4], in the last decade dozens of different heuristic algorithms were proposed to learn structure from the data (e.g., PC [10], Grow-Shrink [9], Sparse Candidate [5], Three Phase Dependency Analysis [3], etc). Unfortunately, most of the algorithms were validated on relatively small networks (e.g., with less than 100 variables) , such as the classical ALARM network [2] or other "toy-networks". Hence, the efficacy of many algorithms for learning real-world networks with thousands of variables remains unknown. The absence of large BNs hinders large-scale learning experiments.

We propose an algorithm for generation of arbitrarily large BNs by tiling smaller real-world known BNs (tiles). The algorithm generates a large network with new edges between the tiles, while preserving joint probability distribution of the tiles. We implemented the algorithm in the BN Tiling Tool available for download from *http://discover1.mc.vanderbilt.edu/discover/public/causal_explorer/* and distributed as a part of Causal Explorer [1].

2. Algorithm

The algorithm requires on input a set of Bayesian networks (tiles) $BN_1, \ldots, BN_n$ and an integer-valued *connectivity parameter* k controlling the number of introduced edges in the output network. The algorithm returns a large Bayesian network consisting of tiles with the new edges between tiles. The joint probability distribution of the tiles is preserved.

Let us denote with Φ_i the variable set of tile i, BN_i the original network giving rise to tile i, $P(\Phi_i)$ the marginal joint probability of the variables Φ_i, and with $P'(\Phi_i)$ the joint probability of BN_i. The constraint we imposed on each tile is that the joint distribution of the variables in the tile remains the same as in the original network the tile is originated from, i.e., $P(\Phi_i) = P'(\Phi_i)$. This is not trivial since edges have been added among the tiles, which may change $P(\Phi_i)$.

The algorithm for generating tiled networks where the above constraint is observed and random edges are added among the tiles is presented in Figure 1. The general idea is the following: edges are added to a node $T \in \Phi_i$ (i.e., T will become a child of the nodes from other tiles) only if T is a minimum node (i.e., has no parents) in BN_i. Then, if for all such T we modify the marginal of T to be the same as the prior of T in the originating network, $P(\Phi_i) = P'(\Phi_i)$ for all i.

In practice we first randomly arrange tiles in topological levels. To ensure that our resulting graph is acyclic we allow connections only between higher and lower topological levels of tiles. We introduce connections between minimal node(s) $T \in \Phi_i$ of the tile i located at level $p \geq 2$ with nodes in ancestor tiles (i.e., located at levels $< p$). For every minimum node $T \in \Phi_i$ we select up to k ancestor nodes. We refer to k as *connectivity parameter*.

Consider example in Figure 2. The basis for the construction of resulting BN is the tile of Asia Bayesian network. The final BN consists of four tiles of Asia. Minimum nodes in the tiles at level 2 receive connections from level 1, and minimum nodes in the tile at level 3 receive connections from levels 1 and 2. The *connectivity parameter* $k = 2$ in this example. The new edges are drawn with dash-lines in Figure 2.

Suppose that a node T receives two new edges from nodes X and Y (See Figure 2). We require that the prior of T in the originating network is the same as the marginal of T in the new network, i.e., $P(T) = P'(T)$ or,

$$P'(T = t) = \sum_{x,y} P(T = t | X = x, Y = y) P(X = x, Y = y)$$

TileBNs$(BN_1 =< \Phi_1, E_1, J_1 >, \ldots, BN_n =< \Phi_n, E_n, J_n >, k)$
Generate $BN =< \Phi, E, J >$ as follows:
1 $\Phi = \cup_i \Phi_i$
2 $E = \cup_i E_i$
3 Randomly arrange tiles in topological levels
4 For every tile i located at level $l \geq 2$
5 For every minimal node $T \in \Phi_i$ from the tile i
6 For every ancestor tile m located at level $< l$
7 Randomly select $p \leq k$ nodes $X_1, \ldots, X_p \in \Phi_m$ from tile m
8 Add new edges $E = E \cup \{X_1 \rightarrow T, \ldots, X_p \rightarrow T\}$
9 For all variables T that received new edges
10 Select randomly x'_{tp}, for all values t of T and all instantiations p
11 of parents of T (i.e., nodes $X_1, \ldots, X_p$), $Pa(T)$
12 Solve equations
13 $\sum_p a_p x'_{tp} r_t c_p = b_t, \forall t$
14 $\sum_t x'_{tp} r_t c_p = 1, \forall p$
15 subject to $r_t c_p \geq 0, \forall t, p$
16 where $a_p = P(Pa(T) = p, b_t = P(T = t)$
17 Set $P(T = t|Pa(T) = p) = x'_{tp} r_t c_p$
18 Let J be the joint implied by the conditional probability tables
19 in $J_1, \ldots, J_n$ and the changes made at line 17
20 Return $BN =< \Phi, E, J >$

FIGURE 1. The algorithm **TileBNs** for tiling BNs in a way that maintains their probabilistic properties.

for all possible values t, x, y of the variables. Equivalently,

$$P'(T = t) = \sum_p P(T = t|Pa(T) = p)P(Pa(T) = p)$$

for $Pa(T)$ being the new parents of T in the simulated BN, and p running through all possible instantiations of the parents. The quantities $P'(T = t)$ are the known priors of T in the originating network, and $P(Pa(T) = p)$ is the joint of $P(Pa(T) = p)$ in the new network. Since our method will produce networks where the constraint $P(\Phi_i) = P'(\Phi_i)$ holds for all i, then $P(Pa(T) = p) = P'(Pa(T) = p)$ and can be exactly calculated from the originating network.

Let us denote with $a_p = P(Pa(T) = p)$ for all different instantiations p of parents of T, with $b_t = P'(T = t)$ for all values of T, and with $x_{tp} = P(T = t|Pa(T) = p)$ for all values of T and parents of T. Then,

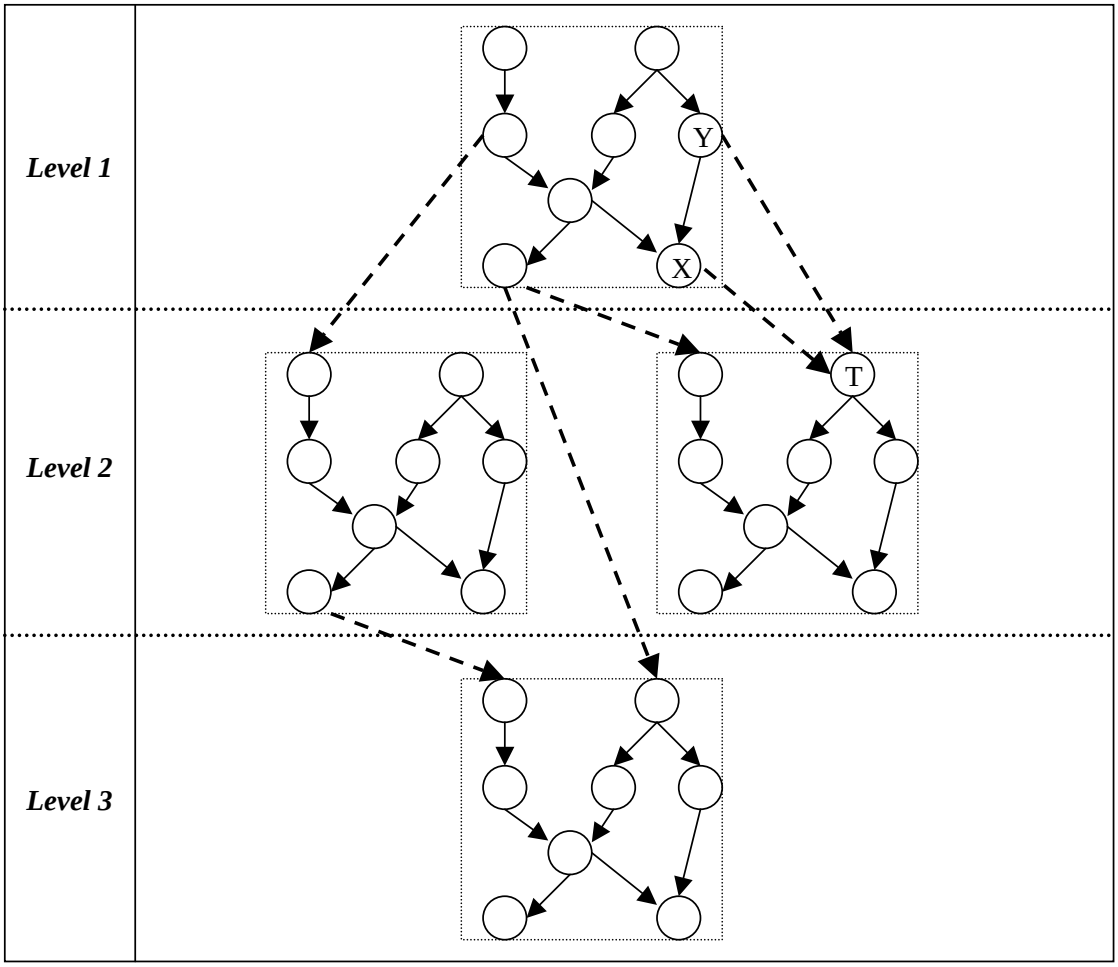

FIGURE 2. Generation of BN from four tiles of Asia network.

the above equations can be rewritten as:

$$\sum_p a_p x_{tp} = b_t, \forall t$$

Also, for x_{tp} to be conditional probabilities they have to belong in $[0, 1]$ and:

$$\sum_t x_{tp} = 1, \forall p$$

must hold for any p. This gives us a set of $|T| + |Pa(T)|$ equations and $|T| \times |Pa(T)|$ unknowns x_{tp} ($|X|$ means cardinality, i.e., number of unique instantiations, of the set of variables X). One could solve this

underconstraint system of linear equations for x_{tp}. However, it is desirable that these solutions are randomly picked among all possible solutions. Most linear equation solvers will arbitrarily, *but not randomly*, select a solution for the system (e.g. solve for the first $|T| + |Pa(T)|$ values of the unknowns and set the rest to zeros).

To find a random solution to the system of equations we took the following approach. We randomly selected values x'_{tp} uniformly from $[0, 1]$. Obviously, the random values will not be solutions to the equations. However, if they are appropriately rescaled they can be. We expressed the rescaling factors as the unknown quantities r_t and c_p and rewrote the equations as:

$$\sum_p a_p x'_{tp} r_t c_p = b_t, \forall t$$

$$\sum_t x'_{tp} r_t c_p = 1, \forall p$$

i.e., we replaced each unknown quantity x_{tp} with the quantity $x'_{tp} r_t c_p$. We also need to introduce constraints:

$$r_t c_p \geq 0, \forall t, p$$

to ensure that $x_{tp} = x'_{tp} r_t c_p \geq 0$ (x_{tp} is automatically ≤ 1 since $\sum_t x_{tp} = 1, \forall p$); hence x_{tp} can be interpreted as probabilities. Now the linear system of $|T| + |Pa(T)|$ equations and $|T| \times |Pa(T)|$ unknowns has become a quadratic system of $|T| + |Pa(T)|$ equations, $|T| + |Pa(T)|$ unknowns (the quantities r_t and c_p), and $|T| \times |Pa(T)|$ nonlinear constraints[1] which can be solved by any standard method for solving non-linear constraint equations.

3. Implementation

BN Tiling algorithm (BN Tiling Tool) was implemented in Mathworks Matlab [7]. The input BNs are specified in HUGIN [8] format, and a simple parser was written to read HUGIN BNs in a custom Matlab format. We used an iterative solver from Matlab Optimization toolbox employing Gauss-Newton algorithm with BFGS updating scheme. Since this optimization algorithm may be trapped in a local minimum, we repeated lines $10 - 16$ in Figure 1 while convergence was not achieved to ensure that the optimization problem is solved numerically with specified tolerance $(1e - 6)$.

[1]For simplicity, one can also substitute specified constraints with linear more strict constraints $|T| + |Pa(T)|$; namely $r_t \geq 0, \forall t$ and $c_p \geq 0, \forall p$

References

[1] C. Aliferis, I. Tsamardinos, and A. Statnikov. Causal explorer: A probabilistic network learning toolkit for biomedical discovery. *International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS)*, 2003.

[2] I. Beinlich, J. Suermondt, R. Chavez, and G. Cooper. The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks. *In Proc. of the Second European Conference on Artificial Intelligence in Medicine*, 1989.

[3] J. Cheng, R. Greiner, J. Kelly, D. Bell, and W. Liu. Learning bayesian networks from data: an information-theory based approach. *The Artificial Intelligence Journal, Volume 137*, 2002.

[4] David Maxwell Chickering. Learning Bayesian networks is NP-Complete. In D. Fisher and H.J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag, 1996.

[5] N. Friedman, I. Nachman, and D. Peer. Learning bayesian network structure from massive datasets: The "sparse candidate" algorithm. *In Proc. 15th Conf. on Uncertainty in Artificial Intelligence*, 1999.

[6] D. Heckerman. A tutorial on learning bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, March 1995.

[7] The Mathworks Inc. Matlab. http://www.mathworks.com, 2003.

[8] F. Jensen, U. Kjærulff, M. Lang, and A.L. Madsen. Hugin - the tool for bayesian networks and influence diagrams. In *First European Workshop on Probabilistic Graphical Models*, pages 212–221, 2002.

[9] D. Margaritis and S. Thrun. Bayesian network induction via local neighborhoods. *Advances in Neural Information Processing Systems 12 (NIPS)*, 1999.

[10] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search, Second Edition*. MIT Press, 2000.

Appendix A. BN Tiling Tool User's Manual

A.1. **System Requirements.** The software runs on a Windows machine, although a UNIX version can be created upon request. It is necessary to have the latest Matlab (e.g., version 6.5 - release 13) with Matlab Optimization Toolbox (e.g., version 2.2 - release 13) and HUGIN software (e.g. version 6.1 or 6.2 educational) installed on your computer.

A.2. **Installation Package.** The software is distributed as DLL's for use with Matlab. The distributive of BN Tiling Tool includes the following files:

- bn_tiling.dll – BN Tiling Tool (Main program) DLL;
- bn_tiling.m – BN Tiling Tool: Matlab driver with documentation;
- data_converter_hugin.dll – Data Converter from HUGIN to Matlab format DLL;
- data_converter_hugin.m – Data Converter From HUGIN to Matlab format: Matlab driver with documentation;
- simulate_data.dll – Data and Adjacency Matrix Generator DLL;
- simulate_data.m – Data and Adjacency Matrix Generator: Matlab driver with documentation;
- Data/alarm_h.dat – ALARM network: data generated by HUGIN;
- Data/alarm_h.mat – ALARM network: data in Matlab format;
- Data/alarm_h.net – ALARM network: original HUGIN network file;
- Data/alarm_h_graph.mat – ALARM network: Adjacency matrix;
- Data/hailfinder_h.dat – Hailfinder network: data generated by HUGIN;
- Data/hailfinder_h.mat – Hailfinder network: data in Matlab format;
- Data/hailfinder_h.net – Hailfinder network: original HUGIN network file;
- Data/hailfinder_h_graph.mat – Hailfinder network: Adjacency matrix;
- Data/nodes.mat – Sample tiled network (about 1000 nodes) from ALARM and Hailfinder tiles.

A.3. **Getting Started.** Given a network file(s) in HUGIN format, one has to perform the following steps in order to generate a large Bayesian network and randomly sample data instance from it:

- Generate data for all original networks using HUGIN software (without missing values). The data will be used to estimate joint probabilities in the tiles. In the later versions of BN Tiling Tool there will be not need to specify datasets, since an inference algorithm will be implemented in the software;
- Convert data generated by HUGIN in Matlab format using utility `data_converter_hugin` and save it in files (a file per single data array variable "data");
- Use `bn_tiling` program to generate a new large Bayesian network;
- Use `sample_data` utility to sample data from the large network, as well as to generate an adjacency matrix.

All details (including examples) on working with functions listed above are included in Matlab documentation of the installation package.