Improved Greedy Algorithms for Learning Graphical Models

Avik Ray, Sujay Sanghavi, *Member, IEEE*, and Sanjay Shakkottai, *Fellow, IEEE*

***Abstract*— We propose new greedy algorithms for learning the structure of a graphical model of a probability distribution, given samples drawn from the distribution. While structure learning of graphical models is a widely studied problem with several existing methods, greedy approaches remain attractive due to their low computational cost. The most natural greedy algorithm would be one which, essentially, adds neighbors to a node in sequence until stopping; it would do this for each node. While it is fast, simple and parallel, this naive greedy algorithm has the tendency to add non-neighbors that show high correlations with the given node. Our new algorithms overcome this problem in three different ways. The recursive greedy algorithm iteratively recovers the neighbors by running the greedy algorithm in an inner loop, but each time only adding the last added node to the neighborhood set. The second forward–backward greedy algorithm includes a node deletion step in each iteration that allows non-neighbors to be removed from the neighborhood set which may have been added in the previous steps. Finally, the greedy algorithm with pruning runs the greedy algorithm until completion and then removes all the incorrect neighbors. We provide both analytical guarantees and empirical performance for our algorithms. We show that in graphical models with strong non-neighbor interactions, our greedy algorithms can correctly recover the graph, whereas the previous greedy and convex optimization-based algorithms do not succeed.**

***Index Terms*— Graphical models, greedy algorithms, forward-backward algorithms, conditional entropy.**

I. Introduction

GRAPHICAL models have been widely used to tractably capture dependence relations amongst a collection of random variables in a variety of domains, ranging from statistical physics, social networks to biological applications [2]–[7]. While hard in general [8], there has been much progress [10]–[12], [14]–[16] in recent years in understanding the relations between the sample and computational complexity for learning the dependence graph between the random variables.

Manuscript received January 20, 2014; revised March 5, 2015; accepted March 29, 2015. Date of publication April 28, 2015; date of current version May 15, 2015. This work was supported in part by NSF under Grant 0954059, Grant 1302435, and Grant CNS-1320175, in part by ARO under Grant W911NF-11-1-0265 and Grant W911NF-14-1-0387, and in part by the DTRA YIP Award. This paper was presented at the 50th Annual Allerton Conference on Communication, Control, and Computing in 2012 [1].

The authors are with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78712 USA (e-mail: avik@utexas.edu; sanghavi@mail.utexas.edu; shakkott@austin.utexas.edu).

Communicated by N. Cesa-Bianchi, Associate Editor for Pattern Recognition, Statistical Learning and Inference.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2015.2427354

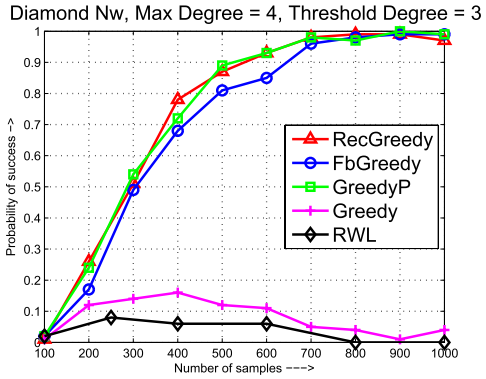

Fig. 1. Performance of different algorithms in an Ising model on diamond network with 6 nodes (Figure 4 with $D = 4$, edge weight $\theta = .5$). Both the $Greedy(\epsilon)$ [14] and RWL [10] algorithms estimate an incorrect edge between nodes 0 and 5 therefore never recovers the true graph G, while our new $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$, $GreedyP(\epsilon)$ algorithms succeed.

We propose three new greedy algorithms to find the Markov graph for any discrete graphical model. While greedy algorithms (that learn the structure by sequentially adding nodes and edges to the graph) tend to have low computational complexity, they are known to fail (i.e., do not determine the correct graph structure) in loopy graphs with low girth [14], even when they have access to exact statistics. This is because a non-neighbor can be the best node (having strong correlation) at a particular iteration; once added, it will always remain. Convex optimization based algorithms like in [10] by Ravikumar et al. (henceforth we call this the RWL algorithm) also cannot provide theoretical guarantees of learning in these situations. These methods require strong incoherence conditions to guarantee success. But such conditions may not be satisfied even in simple graphs [16].

Example: If we run the existing algorithms for an Ising model on a diamond network (Figure 4) with $D = 4$ the performance plot in Figure 1 shows that greedy [14] and RWL [10] algorithms fail to learn the correct graph even with large number of samples.

A. Main Contributions

In this paper, we present three algorithms that overcome this shortfall of greedy and convex optimization based algorithms.

- The *recursive greedy algorithm* is based on the observation that the *last* node added by the simple, naive greedy algorithm is always a neighbor; thus, we can use the naive greedy algorithm as an inner loop that, after every execution, yields just one more neighbor (instead of the entire set).

- The *forward-backward greedy algorithm* takes a different tack, interleaving node addition (forward steps) with node removal (backward steps). In particular, in every iteration, the algorithm looks for nodes in the existing set that have a very small marginal effect; these are removed. Inclusion of correct neighbors reduces the influence of non-neighbors, and in the end all such spurious nodes get removed leaving us with the correct neighborhood set.
- Our third algorithm, namely the *greedy algorithm with pruning*, first runs the greedy algorithm until it is unable to add any more nodes to the neighborhood estimate. Subsequently, it executes a node pruning step that identifies and removes all the incorrect neighbors that were possibly included in the neighborhood estimate by the greedy algorithm.
- We show that all three algorithms can efficiently learn the structure of a large class of graphical models even without correlation decay, and when the graph has strong correlation between non-neighbors. We calculate the sample complexity and computational (number of iterations) complexity for these algorithms (with high probability) under a natural non-degeneracy assumption (Theorems 1 and 2);
- Finally we present numerical results that indicate tractable sample and computational complexity for loopy graphs (diamond graph, grid).

B. Related Work

Several approaches have been taken so far to learn the graph structure of MRF in presence of cycles.

First, search based algorithms like local independence test (LIT) by Bresler et al. in [11], conditional variation distance thresholding (CVDT) by Anandkumar et al. in [15] and conditional independence test by Wu et al. [12] try to find the smallest set of nodes through exhaustive search, conditioned on which either a given node is independent of rest of the nodes in the graph, or a pair of nodes are independent of each other. These algorithms have a fairly good sample complexity, but due to exhaustive search they have a high computation complexity. Also to run these algorithms one needs to know some additional information about the graph structure. For example the local independence test needs the knowledge of the maximum degree of the graph and the CVDT algorithm needs the knowledge of the maximum size of the local separator for the graph.

Second, an algorithm with very good sample complexity using convex optimization was proposed (for Ising models) in [10] by Ravikumar et al. This was further generalized for any pairwise graphical model in [13]. These algorithms try to construct a pseudo likelihood function using the parametric form of the distribution such that it is convex and try to maximize it over the parameter values. The optimized parameter values in effect reveal the Markov graph structure. However these algorithms require a strong incoherence assumption to guarantee its success. In [16], Bento and Montanari showed that even for a large class of Ising models, the incoherence conditions are not satisfied hence the convex optimization based algorithms fail. They also show that in Ising models with weak long range correlation, a simple low complexity thresholding algorithm can correctly learn the graph.

Finally, a greedy learning algorithm was proposed in [14] which tries to find the minimum value of the conditional entropy of a particular node in order to estimate its neighborhood. We call this algorithm as $Greedy(\epsilon)$. It is an extension of the Chow and Liu [9] algorithm to graphs with cycles. It was shown that for graphs with correlation decay and large girth this exactly recovers the graph G. However it fails for graphs with large correlation between non-neighbors. A forward-backward greedy algorithm based on convex optimization was also presented recently by Jalali et al. in [17], which works for any pairwise graphical model. This required milder assumption than in [10] and also gives a better sample complexity.

This paper is organized as follows. First we review the definition of a graphical model and the graphical model learning problem in section II. The three greedy algorithms are described in section III. Next we give sufficient conditions for the success of the greedy algorithms in section IV. In section V we present the main theorems showing the performance of the recursive greedy, forward-backward and greedy with pruning algorithms. We compare the performance of our algorithm with other well known algorithms in section VI. In section VII we present some simulation results. The proofs are presented in the appendix.

II. Brief Review: Graphical Models

In this section we briefly review the general graphical model and the Ising model [10], [18]. Let $X = (X_1, X_2, \ldots, X_p)$ be a random vector over a discrete set $\mathcal{X}^p$, where $\mathcal{X} = \{1, 2, \ldots, m\}$. $X_S = (X_i : i \in S)$ denote the random vector over the subset $S \subseteq \{1, 2, \ldots, p\}$. Let $G = (V, E)$ denote a graph having p nodes. Let Δ be the maximum degree of the graph G and Δ_i be the degree of the i^{th} node. An undirected graphical model or Markov random field is a tuple $M = (G, X)$ such that each node in G corresponds to a particular random variable in X. Moreover G captures the Markov dependence between the variables X_i such that absence of an edge (i, j) implies the conditional independence of variables X_i and X_j given all the other variables.

For any node $r \in V$, let $\mathcal{N}_r$ denote the set of neighbors of r in G. Then the distribution $\mathbb{P}(X)$ has the special Markov property that for any node r, X_r is conditionally independent of $X_{V \setminus \{r\} \bigcup \mathcal{N}_r}$ given $X_{\mathcal{N}_r} = \{X_i : i \in \mathcal{N}_r\}$, the neighborhood of r, i.e.

$$\mathbb{P}(X_r | X_{V \setminus r}) = \mathbb{P}(X_r | X_{\mathcal{N}_r}) \tag{1}$$

A graphical model is called an **Ising model** when the joint distribution of $\{X_i\}$ has only pairwise interactions and take values in the set $\mathcal{X} = \{-1, 1\}$. For this paper we also consider the node potentials as zero (the zero field Ising model). Hence the distribution take the following simplified form.

$$\mathbb{P}_\Theta(X = x) = \frac{1}{Z} \exp\left\{ \sum_{(i,j) \in E} \theta_{ij} x_i x_j \right\} \tag{2}$$

TABLE I

PERFORMANCE COMPARISON BETWEEN DIFFERENT DISCRETE GRAPHICAL MODEL LEARNING ALGORITHMS IN LITERATURE. Δ DENOTES THE MAXIMUM DEGREE OF THE GRAPH AND p IS THE NUMBER OF NODES. $\mathcal{X}$ IS THE ALPHABET SET FROM WHICH A RANDOM VARIABLE TAKE ITS VALUE IN THE DISCRETE GRAPHICAL MODEL. ϵ IS A NON-DEGENERACY PARAMETER FOR THE GRAPHICAL MODEL (SEE CONDITION (A1) IN SECTION IV). α IS AN INPUT PARAMETER TO DETERMINE THE ELIMINATION THRESHOLD IN *FbGreedy* ALGORITHM (SEE SECTION III-B). C_{min} IS A DEPENDENCY PARAMETER IN [10]. θ IS THE EDGE WEIGHT PARAMETER OF THE ISING MODEL IN [16]

Algorithm	Graph family / Assumptions	Sample complexity	Computation complexity
Bresler et al. [11]	Δ degree limited, non-degenerate	$\Omega\left(\lvert\mathcal{X}\rvert^{4\Delta}\Delta\log p\right)$	$O\left(p^{2\Delta+1}\log p\right)$
CVDT [15]	$(\Delta,\gamma)-$ local separation	$\Omega\left(\lvert\mathcal{X}\rvert^{2\Delta}(\Delta+2)\log p\right)$	$O\left(p^{\Delta+2}\right)$
RWL [10]	Δ degree limited, Ising model, dependency, incoherence	$\Omega\left(\frac{\Delta^3}{C_{min}^2}\log p\right)$	$O\left(p^4\right)$
Jalali et al. [13]	Δ degree limited, pairwise graphical model, RSC	$\Omega\left(\Delta^2\log p\right)$	$O\left(p^4\right)$
Greedy [14]	Δ degree limited, large girth, correlation decay, non-degenerate	$\Omega\left(\lvert\mathcal{X}\rvert^{4\Delta}\log p\right)$	$O\left(p^2\Delta\right)$
RecGreedy	Δ degree limited, non-degenerate	$\Omega\left(\frac{\lvert\mathcal{X}\rvert^{2\log\lvert\mathcal{X}\rvert/\epsilon}}{\epsilon^5}\log p\right)$	$O\left(p^2\frac{\Delta}{\epsilon}\right)$
FbGreedy	Δ degree limited, non-degenerate	$\Omega\left(\frac{\lvert\mathcal{X}\rvert^{4\log\lvert\mathcal{X}\rvert/((1-\alpha)\epsilon)}}{\epsilon^5(1-\alpha)\alpha^4}\log p\right)$	$O\left(\frac{p^2}{(1-\alpha)\epsilon}\right)$
GreedyP	Δ degree limited, non-degenerate	$\Omega\left(\frac{\lvert\mathcal{X}\rvert^{2\log\lvert\mathcal{X}\rvert/\epsilon}}{\epsilon^5}\log p\right)$	$O\left(\frac{p^2}{\epsilon}\right)$
Bento and Montanari [16]	Δ degree limited, Ising model, correlation decay	$\Omega\left(\frac{\Delta^2}{(1-2\Delta\tanh\theta)^2}\log p\right)$	$O\left(p^2\right)$

where $x_i, x_j \in \{-1, 1\}$, $\theta_{ij} \in \mathbb{R}$ and Z is the normalizing constant.

The graphical **model selection problem** is as follows. Given n independent samples $\mathcal{S}_n = \{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\}$ from the distribution $\mathbb{P}(X)$, where each $x^{(i)}$ is a p dimensional vector $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_p^{(i)}) \in \{1, \ldots, m\}^p$, the problem is to estimate the Markov graph G corresponding to the distribution $\mathbb{P}(X)$ by recovering the correct edge set E. This problem is NP hard and has been solved only under special assumptions on the graphical model structure. In some cases a learning algorithm is able to find the correct neighborhood of each node $v \in V$ with a high probability and hence recover the true topology of the graph. Table I has a brief survey of some relevant methods for discrete graphical models; the algorithms RecGreedy, FbGreedy and GreedyP are proposed in this paper (see Section III for algorithm descriptions).

We describe some notations. For a subset $S \subseteq V$, we define $P(x_S) = \mathbb{P}(X_S = x_S)$, $x_S \in \mathcal{X}^{|S|}$. The empirical distribution $\widehat{P}(X)$ is the distribution of X computed from the samples. Let $i \in V - S$, the entropy of the random variable X_i conditioned on X_S is written as $H(X_i|X_S)$. The empirical entropy calculated corresponding to the empirical distribution $\widehat{P}$ is denoted by $\widehat{H}$. If P and Q are two probability measures over a finite set $\mathcal{Y}$, then the total variational distance between them is given by, $||P - Q||_{TV} = \frac{1}{2}\sum_{y\in\mathcal{Y}} |P(y) - Q(y)|$.

III. GREEDY ALGORITHMS

In this section we describe three new greedy algorithms for learning the structure of a MRF. First we recall the recent greedy algorithm proposed by Netrapalli it et al. [14]. The $Greedy(\epsilon)$ algorithm finds the neighborhood of node i by greedily adding nodes to the set $\widehat{N}(i)$ which causes the most decrease in conditional entropy $\widehat{H}(X_i|X_{\widehat{N}(i)})$. When no more node addition is possible the set $\widehat{N}(i)$ gives the neighborhood estimate of node i. The high level pseudo-code is shown in Algorithm 1.

The naive $Greedy(\epsilon)$ algorithm suffers from the problem that in Step 4 node j can be a non-neighbor which produces the most decrease in conditional entropy, hence gets added to neighborhood set $\widehat{N}(i)$. We now describe our new greedy algorithms which overcome this problem using three different techniques.

Algorithm 1 $Greedy(\epsilon)$ [14]

1: **for** $i \in V$ **do**
2: $\quad\widehat{N}(i) \leftarrow \phi$
3: $\quad$**do**
4: $\quad\quad$Find node $j \in V\backslash\widehat{N}(i)$ for which $\delta_j = \widehat{H}(X_i|X_{\widehat{N}(i)}) - \widehat{H}(X_i|X_{\widehat{N}(i)}, X_j)$ is maximized. Add node j to $\widehat{N}(i)$ if $\delta_j \geq \frac{\epsilon}{2}$
5: $\quad$**while** New node has been added to $\widehat{N}(i)$
6: **end for**
7: Output edge set $E = \{(i, j) : i \in \widehat{N}(j)$ and $j \in \widehat{N}(i)\}$

A. Recursive Greedy Algorithm

Idea: Consider first the simpler setting when we know the exact distribution $\mathbb{P}(X)$. The naive greedy algorithm (Algorithm 1) adds nodes to the neighborhood set $\widehat{N}(i)$, stopping when no further strict reduction in conditional entropy $H(X_i|X_{\widehat{N}(i)})$ is possible. This stopping occurs when the true neighborhood $\mathcal{N}_i$ is a subset of the estimated neighborhood $\widehat{N}(i)$. Our key observation here is that the *last* node to be added to the set $\widehat{N}(i)$ by the naive greedy algorithm will always be in the true neighborhood $\mathcal{N}_i$, since inclusion of the last neighbor in the conditioning set enables it to reach the minimum conditional entropy. We leverage this observation as follows. We run the naive greedy algorithm as an inner loop, using a conditioning set $\widehat{T}(i)$; at the end of every run of this inner loop, we pick only the last node added to $\widehat{T}(i)$ and add it to the estimated neighborhood $\widehat{N}(i)$. The next inner loop starts with the set $\widehat{T}(i)$ initialized to the current estimated neighborhood $\widehat{N}(i)$, and it proceeds to find the next neighbor. Hence the algorithm discovers a neighbor in each run of the innermost loop and finds all the neighbors of a given node i in exactly Δ_i iterations of the outer loop.

Algorithm 2 $RecGreedy(\epsilon)$ (High Level)

for $i \in V$ **do**
  $\widehat{N}(i), \widehat{T}(i) \leftarrow \phi$
  do
    Set $\widehat{T}(i) = \widehat{N}(i)$
    Find node $j \in V \backslash \widehat{T}(i)$ for which $\delta_j = \widehat{H}(X_i|X_{\widehat{T}(i)}) - \widehat{H}(X_i|X_{\widehat{T}(i)}, X_j)$ is maximized. Add node j to $\widehat{T}(i)$ if $\delta_j \geq \frac{\epsilon}{2}$. Repeat till no new node can be added to $\widehat{T}(i)$
    Add the node l last added to $\widehat{T}(i)$ in the previous step to the set $\widehat{N}(i)$
  while New node has been added to $\widehat{N}(i)$
end for
Output edge set $E = \{(i, j) : i \in \widehat{N}(j) \text{ and } j \in \widehat{N}(i)\}$

Algorithm 3 $FbGreedy(\epsilon, \alpha)$ (High Level)

for $i \in V$ **do**
  $\widehat{N}(i) \leftarrow \phi$
  do
    Find node $j \in V \backslash \widehat{N}(i)$ for which $\delta_j = \widehat{H}(X_i|X_{\widehat{N}(i)}) - \widehat{H}(X_i|X_{\widehat{N}(i)}, X_j)$ is maximized. Add node j to $\widehat{N}(i)$ if $\delta_j \geq \frac{\epsilon}{2}$
    Find node $l \in \widehat{N}(i)$ for which $\gamma_l = \widehat{H}(X_i|X_{\widehat{N}(i)\backslash l}) - \widehat{H}(X_i|X_{\widehat{N}(i)})$ is minimized. Remove node l from $\widehat{N}(i)$ if $\gamma_l \leq \frac{\alpha\epsilon}{2}$
  while Node is added to or removed from $\widehat{N}(i)$
end for
Output edge set $E = \{(i, j) : i \in \widehat{N}(j) \text{ and } j \in \widehat{N}(i)\}$

Algorithm 4 $GreedyP(\epsilon)$ (High Level)

for $i \in V$ **do**
  Run $Greedy(\epsilon)$ to generate neighborhood estimate $\widehat{N}(i)$
  For each $j \in \widehat{N}(i)$ compute $\gamma_j = \widehat{H}(X_i|X_{\widehat{N}(i)\backslash j}) - \widehat{H}(X_i|X_{\widehat{N}(i)})$. If $\gamma_j \leq \frac{\epsilon}{2}$ remove node j from $\widehat{N}(i)$
end for
Output edge set $E = \{(i, j) : i \in \widehat{N}(j) \text{ and } j \in \widehat{N}(i)\}$

The above idea works as long as every neighbor has a measurable effect on the conditional entropy, even when there are several other variables in the conditioning. The algorithm is $RecGreedy(\epsilon)$, high level pseudocode detailed below. It needs a non-degeneracy parameter ϵ, which is the threshold for how much effect each neighbor has on the conditional entropy. A more detailed pseudo-code is given in Appendix B.

Note that in all our greedy algorithms, in order to maintain edge consistency in the estimated graph, we output an edge (i, j) if $i \in \widehat{N}(j)$ and $j \in \widehat{N}(i)$. This may lead to spurious or missing edges when individual neighborhoods are incorrectly estimated. However we prove that the likelihood of such errors tend to zero asymptotically with large number of samples (Theorem 1).

B. Forward-Backward Greedy Algorithm

Our second algorithm takes a different approach to fix the problem of spurious nodes added by the naive greedy algorithm, by adding a backward step at every iteration that prunes nodes it detects as being spurious. In particular, after every forward step that adds a node to the estimated neighborhood $\widehat{N}(i)$, the algorithm finds the node in this new estimated neighborhood that has the smallest individual effect on the new conditional entropy; i.e. removing this node from $\widehat{N}(i)$ will produce the least increase in conditional entropy. If this increase is too small, this node is removed from the estimated neighborhood $\widehat{N}(i)$.

The algorithm, $FbGreedy(\epsilon, \alpha)$, is described below (detailed pseudo-code in Appendix B). It takes two input parameters beside the samples. The first is the same non-degeneracy parameter ϵ as in the $RecGreedy(\epsilon)$ algorithm. The second parameter $\alpha \in (0, 1)$ is utilized by the algorithm to determine the threshold of elimination in the backward step. We will see later that this parameter also helps to trade-off between the sample and computation complexity of the $FbGreedy(\epsilon, \alpha)$ algorithm. The algorithm stops when there are no further forward or backward steps.

C. Greedy Algorithm With Pruning

The third algorithm overcomes the problem of non-neighbor inclusion in the $Greedy(\epsilon)$ algorithm by adding a node pruning step after the execution of the greedy algorithm (similar to the backward step in $FbGreedy(\epsilon, \alpha)$). In this algorithm, after running the $Greedy(\epsilon)$ algorithm, the pruning step declares a neighbor node to be spurious if its removal from the neighborhood estimate $\widehat{N}(i)$ does not significantly increase the final conditional entropy. These spurious nodes are removed to result in an updated neighborhood estimate for each node.

The pseudocode of this greedy algorithm with node-pruning – $GreedyP(\epsilon)$ – is given in Algorithm 4. In addition to the samples, the input is again a non-degeneracy parameter ϵ similar to $RecGreedy(\epsilon)$ and $FbGreedy(\epsilon, \alpha)$ algorithms.

D. Choice of Parameters

We now briefly describe how the different parameters ϵ, α are chosen in the greedy algorithms. For real datasets the non-degeneracy parameter ϵ can be chosen through cross validation over a labeled training/held out dataset as the value for which a suitable performance metric is maximized (i.e. the model best fits the given data). Other graphical model learning algorithms require similar parameters e.g. regularization parameter λ in RWL [10] and local separator size η in CVDT algorithm [15] for recovery. In Figure 2 we plot the cross validation profile of the greedy algorithms on synthetic dataset with increasing values of ϵ, and probability of success as the performance metric. We can see that the greedy algorithms correctly recover the graph G for a wide range of ϵ hence is robust to the choice of the parameter. The parameter α in $FbGreedy$ algorithm can be chosen as any value in $(0, 1)$. We choose $\alpha = .9$ in our experiments.

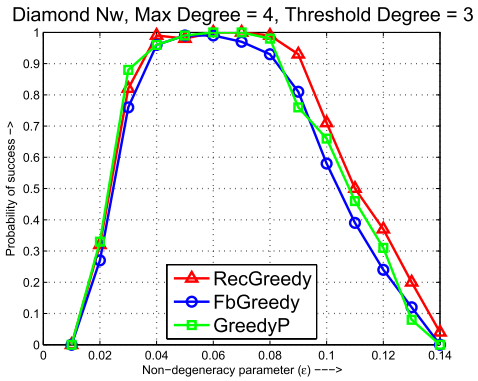

Fig. 2. Figure showing the cross validation profile of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms for Ising model in a diamond network with $D = 4, \theta = .5$ and $n = 1000$ samples. The greedy algorithms can correctly recover the graph for a wide range of choice of ϵ.

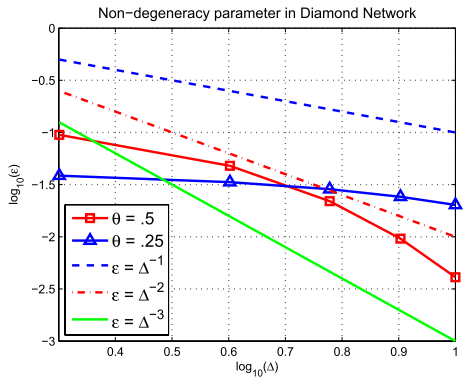

Fig. 3. Figure showing the variation of non-degeneracy parameter ϵ for an Ising model over a diamond network with increasing maximum degree Δ. For smaller values of edge weight parameter θ the non-degeneracy parameter approximately scales as $\epsilon \approx \frac{c}{\Delta}$ for some constant c.

IV. Sufficient Conditions for Markov Graph Recovery

In this section we describe the sufficient condition which guarantees that the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms recover the correct Markov graph G.

A. Non-Degeneracy

Our non-degeneracy assumption requires every neighbor have a significant enough effect. Other graphical model learning algorithms require similar assumptions to ensure correctness [10], [11], [14].

(A1) Non-Degeneracy Condition: Consider the graphical model $M = (G, X)$, where $G = (V, E)$. Then there exists $\epsilon > 0$ such that for all $i \in V$, $A \subset V$, $\mathcal{N}_i \not\subset A$ and $j \in \mathcal{N}_i$, $j \notin A$ the following condition holds.

$$H(X_i|X_A, X_j) < H(X_i|X_A) - \epsilon \tag{3}$$

Thus by adding a neighboring node to any conditioning set that does not already contain it, the conditional entropy strictly decreases by at least ϵ. Also the above condition together with the local Markov property (1) implies that the conditional entropy attains a unique minimum at $H(X_i|X_{\mathcal{N}_i})$. Note that condition (A1) can also be written as $I(X_i; X_j|X_A) > \epsilon$, where $I(.)$ is the mutual information. Expressed in this form the non-degeneracy condition has an alternative interpretation as follows. Conditioned on any set A which does not contain all the neighbors, a neighbor j of node i has a non-zero information about the distribution of X_i. Condition (A1) is also necessary for the success of search based algorithms e.g. CMIT in [15], which exploit the global Markov property to recover graph G. This is because when (A1) is not satisfied there exists a set A such that $I(X_i; X_j|X_A) = 0$, hence this set A is detected as a separator between nodes i and j, therefore j is not included in the neighborhood of node i.

In Figure 3 we plot how the non-degeneracy parameter ϵ scales in an Ising model over a diamond network (Figure 4) with varying maximum degree. When the edge weight parameter θ of the Ising model is small ϵ approximately scales linearly in $\frac{1}{\Delta}$. In Section V we will show that for any graphical model satisfying non-degeneracy condition (A1) the non-degeneracy parameter ϵ is also upper bounded as $\epsilon = O\left(\frac{1}{\Delta}\right)$.

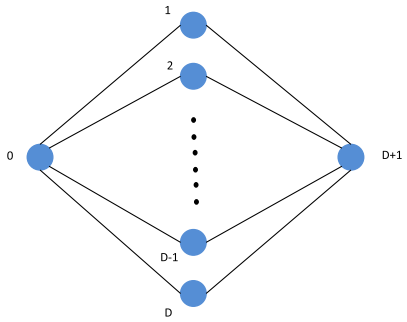

Fig. 4. An example of a diamond network with $D+2$ nodes and maximum degree D where $Greedy(\epsilon)$ can fail but $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms always correctly recover the true graph.

V. Main Result

In this section we state our main result showing the performance of the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms. First we state some useful lemmas. We restate the first lemma from [14] and [19] that will be used to show the concentration of the empirical entropy $\widehat{H}$ with samples.

Lemma 1: Let P and Q be two discrete distributions over a finite set $\mathcal{X}$ such that $||P - Q||_{TV} \leq \frac{1}{4}$. Then,

$$|H(P) - H(Q)| \leq 2||P - Q||_{TV} \log \frac{|\mathcal{X}|}{2||P - Q||_{TV}}$$

The following two lemmas bound the number of steps in $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms which also guarantees their convergence.

Lemma 2: The number of greedy steps in each recursion of the $RecGreedy(\epsilon)$ and in $GreedyP(\epsilon)$ algorithm is less than $\frac{2\log|\mathcal{X}|}{\epsilon}$.

Proof: In each step the conditional entropy is reduced by an amount at least $\epsilon/2$. Since the maximum reduction in entropy possible is $\widehat{H}(X_i) \leq \log|\mathcal{X}|$, the number of steps is upper bounded by $\frac{2\log|\mathcal{X}|}{\epsilon}$. □

Remark: Note that the $GreedyP(\epsilon)$ algorithm will take at least Δ steps to include all the neighbors in the conditioning set. Lemma 2 states that the maximum number of steps is upper bounded by $\frac{2\log|\mathcal{X}|}{\epsilon}$. This implies $\frac{2\log|\mathcal{X}|}{\epsilon} \geq \Delta$.

Lemma 3: The number of steps in the $FbGreedy(\epsilon,\alpha)$ is upper bounded by $\frac{4\log|\mathcal{X}|}{\epsilon(1-\alpha)}$.

Proof: Note that as long as the forward step is active (which occurs till all neighbors are included in the conditioning set $\widehat{N}(i)$), in each step the conditional entropy reduces by at least $(1-\alpha)\epsilon/2$. Hence all the neighbors are included within $\frac{2\log|\mathcal{X}|}{(1-\alpha)\epsilon}$ steps. The number of non-neighbors included in the conditioning set is also bounded by $\frac{2\log|\mathcal{X}|}{(1-\alpha)\epsilon}$. Thus it will take at most the same number backward steps to remove the non-neighbors. Hence the total number of steps is at most $\frac{4\log|\mathcal{X}|}{(1-\alpha)\epsilon}$. □

We now state our main theorem showing the performance of Algorithms 2, 3 and 4.

Theorem 1: Consider a MRF over a graph G with maximum degree Δ, having a distribution $P(X)$.

1) Correctness (Non-Random): Suppose (A1) holds and the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon,\alpha)$ and $GreedyP(\epsilon)$ algorithms have access to the true conditional entropies therein, then they correctly estimate the graph G.

2) Sample Complexity: Suppose (A1) holds and $0<\delta<1$.

- *When the number of samples $n = \Omega\left(\frac{|\mathcal{X}|^{2\log|\mathcal{X}|/\epsilon}}{\epsilon^5}\log\frac{p}{\delta}\right)$ the $RecGreedy(\epsilon)$ correctly estimates G with probability greater than $1-\delta$.*
- *When the number of samples $n = \Omega\left(\frac{|\mathcal{X}|^{4\log|\mathcal{X}|/((1-\alpha)\epsilon)}}{\epsilon^5(1-\alpha)\alpha^4}\log\frac{p}{\delta}\right)$ the $FbGreedy(\epsilon,\alpha)$ correctly estimates G with probability greater than $1-\delta$, for $0<\alpha<1$.*
- *When the number of samples $n = \Omega\left(\frac{|\mathcal{X}|^{2\log|\mathcal{X}|/\epsilon}}{\epsilon^5}\log\frac{p}{\delta}\right)$ the $GreedyP(\epsilon)$ correctly estimates G with probability greater than $1-\delta$.*

The proof of correctness with true conditional entropies known is straightforward under non-degenerate assumption (A1). The proof in presence of samples is based on Lemma 4 similar to [14, Lemma 2] showing the concentration of empirical conditional entropy, which is critical for the success of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon,\alpha)$ and $GreedyP(\epsilon)$ algorithms. We show that with sufficiently many samples the empirical distributions and hence the empirical conditional entropies also concentrate around their true values with a high probability. This will ensure that algorithms 2, 3 and 4 correctly recover the Markov graph G. The complete proof is presented in Appendix A.

Lemma 4: Consider a graphical model $M=(G,X)$ with distribution $P(X)$. Let $0<\delta_3<1$. If the number of samples

$$n > \frac{8|\mathcal{X}|^{2(s+2)}}{\zeta^4}\left[(s+1)\log 2p|\mathcal{X}| + \log\frac{1}{\delta_3}\right]$$

then with probability at least $1-\delta_3$

$$|\widehat{H}(X_i|X_S) - H(X_i|X_S)| < \zeta$$

for any $S\subset V$ such that $|S|\leq s$.

Lemma 4 follows from Lemma 1 and Azuma's inequality. Although the sample complexities of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon,\alpha)$ and $GreedyP(\epsilon)$ algorithms are slightly more than other non-greedy algorithms [10], [11], [15], the main appeal of these greedy algorithms lie in their low computation complexity. The following theorem characterizes the computation complexity of Algorithms 2, 3 and 4. When calculating the run-time, each arithmetic operation and comparison is counted as an unit-time operation. For example to execute line 6 in Algorithm 5, each comparison takes an unit-time and each entropy calculation takes $O(n)$ time (since there are n samples using which the empirical conditional entropy is calculated). Since there are at most $p-1$ comparisons the total time required to execute this line is $O(np)$.

Theorem 2 (Run-Time): Consider a graphical model $M=(G,X)$, with maximum degree Δ, satisfying assumptions (A1). Then the expected run-time of the greedy algorithms are,

- $O\left(\delta\frac{p^3}{\epsilon}n + (1-\delta)\frac{p^2}{\epsilon}\Delta n\right)$ *for the $RecGreedy(\epsilon)$ algorithm.*
- $O\left(\frac{p^2}{(1-\alpha)\epsilon}n\right)$ *for the $FbGreedy(\epsilon,\alpha)$ algorithm.*
- $O\left(\frac{p(p+1)}{\epsilon}n\right)$ *for the $GreedyP(\epsilon)$ algorithm.*

The proofs of Theorem 2 are given in Appendix.

Remark: Note that if we take $\alpha < \frac{\Delta-1}{\Delta}$ the $FbGreedy(\epsilon,\alpha)$ has a better run time guarantee than the $RecGreedy(\epsilon)$ algorithm for small δ. Also for small δ, $GreedyP(\epsilon)$ algorithm has the best runtime among three all greedy algorithms.

VI. Performance Comparison

In this section we compare the performance of the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon,\alpha)$ and $GreedyP(\epsilon)$ algorithms with other graphical model learning algorithms.

A. Comparison With Greedy(ϵ) Algorithm

The $RecGreedy(\epsilon)$, $FbGreedy(\epsilon,\alpha)$ and $GreedyP(\epsilon)$ algorithms are strictly better than the $Greedy(\epsilon)$ algorithm in [14]. This is because Algorithms 2, 3 and 4 always find the correct graph G when the $Greedy(\epsilon)$ finds the correct graph, but they are applicable to a wider class of graphical models since they do not require the assumption of large girth or correlation decay to guarantee its success. Note that $RecGreedy(\epsilon)$, $FbGreedy(\epsilon,\alpha)$ and $GreedyP(\epsilon)$ algorithms use the $Greedy(\epsilon)$ algorithm as an intermediate step. Hence when $Greedy(\epsilon)$ finds the true neighborhood $\mathcal{N}_i$ of node i, $RecGreedy(\epsilon)$ algorithm will find the correct neighborhood in each of the recursive steps and $FbGreedy(\epsilon,\alpha)$, $GreedyP(\epsilon)$ algorithms output the correct neighborhood directly without having to utilize any of the backward steps or the pruning step respectively. Hence $RecGreedy(\epsilon)$, $FbGreedy(\epsilon,\alpha)$ and $GreedyP(\epsilon)$ algorithms also succeed in finding the true graph G. We now demonstrate a clear example of a graph where $Greedy(\epsilon)$ fails to recover the true graph but the Algorithms 2, 3 and 4 are successful. This example is also presented in [14]. Consider an Ising model on the graph in Figure 4. We have the following proposition.

Proposition 1: Consider an Ising model with $V = \{0, 1, \ldots, D, D+1\}$ *and* $E = \{(0,i), (i, D+1)\ \forall i : 1 \leq i \leq D\}$ *with a distribution function* $P(x) = \frac{1}{Z}\prod_{(ij)\in E} e^{\theta x_i x_j}$, $X_i \in \{1, -1\}$. *Then with* $D > \frac{2\theta}{\log\cosh(2\theta)} + 1$ *we have*

$$H(X_0|X_{D+1}) < H(X_0|X_1)$$

The proof follows from straightforward calculation (see Appendix). Hence for the Ising model considered above (Figure 4) with $D > \frac{2\theta}{\log\cosh(2\theta)} + 1$ the $Greedy(\epsilon)$ incorrectly includes node $D+1$ in the neighborhood set in the first step. However with an appropriate ϵ the MRF satisfies assumption $(A1)$. Therefore Theorem 1 ensures that the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms correctly estimate the graph G.

B. Comparison With Search Based Algorithms

Search based graphical model learning algorithms like the Local Independence Test (LIT) by Bresler et al. [11] and the Conditional Variation Distance Thresholding (CVDT) by Anandkumar *et al.* [15] generally have good sample complexity, but high computation complexity. As we will see the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms have slightly more sample complexity but significantly lower computational complexity than the search based algorithms. Moreover to run the search based algorithms one needs to know the maximum degree Δ for LIT and the maximum size of the separator η for the CVDT algorithm. However the greedy algorithms can be run without knowing the maximum degree of the graph.

For bounded degree graphs the LIT algorithm has a sample complexity of $\Omega(|\mathcal{X}|^{4\Delta}\Delta\log\frac{2p}{\delta})$. Without any assumption on the maximum size of the separator, for bounded degree graphs the CVDT algorithm also has a similar sample complexity of $\Omega(|\mathcal{X}|^{2\Delta}(\Delta+2)\log\frac{p}{\delta})$. Note that the quantity P_{min} in the sample complexity expression for CVDT algorithm ([15, Th. 2]) is the minimum probability of $P(X_S = x_S)$ where $|S| \leq \eta + 1$. This scales with Δ as $P_{min} \leq \frac{1}{|\mathcal{X}|^{\eta+1}}$. For general degree bounded graphs we have $\eta = \Delta$. The sample complexity for $RecGreedy(\epsilon)$, $GreedyP(\epsilon)$ and $FbGreedy(\epsilon, \alpha)$ algorithms is slightly higher at $\Omega\left(\frac{|\mathcal{X}|^{2\log|\mathcal{X}|/\epsilon}}{\epsilon^5}\log\frac{p}{\delta}\right)$ and $\Omega\left(\frac{|\mathcal{X}|^{4\log|\mathcal{X}|/((1-\alpha)\epsilon)}}{\epsilon^5(1-\alpha)\alpha^4}\log\frac{p}{\delta}\right)$ respectively (since $\frac{2\log|\mathcal{X}|}{\epsilon} > \Delta$). However the computation complexity of the LIT algorithm is $O(p^{2\Delta+1}\log p)$ and that of the CVDT algorithm is $O(|\mathcal{X}|^{\Delta}p^{\Delta+2}n)$, which is much larger than $O\left(\frac{p^2}{\epsilon}\Delta n\right)$ for $RecGreedy(\epsilon)$ algorithm, $O\left(\frac{p^2}{(1-\alpha)\epsilon}n\right)$ for the $FbGreedy(\epsilon, \alpha)$ algorithm and $O\left(\frac{p^2}{\epsilon}n\right)$ for $GreedyP(\epsilon)$ algorithm.

We finally comment that in [11] the authors showed that under an additional exponential correlation decay assumption, the computation complexity of the LIT algorithm can be decreased by reducing the search space through a correlation-neighborhood selection procedure. Under similar assumptions it can be shown that the computation complexity of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms (and the CVDT algorithm [15]) can also be reduced. Even in this case (i.e., with correlation decay), we can show that the greedy algorithms still outperform the search based algorithms in run-time. However, note that the correlation decay assumption is not necessary to guarantee the success of the greedy algorithms presented in this paper; specifically, the sample complexity and run-time results presented so far do not make this assumption.

C. Comparison With Convex Optimization Based Algorithms

In [10] Ravikumar et al. presented a convex optimization based learning algorithm for Ising models, which we have referred as the RWL algorithm. It was later extended for any pairwise graphical model by Jalali et al. in [13]. These algorithms assume extra incoherence or restricted strong convexity conditions hold, in which case they have a low sample complexity of $\Omega\left(\Delta^3\log p\right)$, when dependency parameter $C_{min} = \Omega(1)$. However these algorithms have a computation complexity of $O(p^4)$ higher than the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms. Moreover the greedy algorithms we propose are applicable for a wider class of graphical models. Finally these optimization based algorithms require a strong incoherence property to guarantee its success; conditions which may not hold even for a large class of Ising models as shown by Bento and Montanari in [16]. They also prove that the RWL algorithm fails in a diamond network (Figure 4) for a large enough degree, whenever there is a strong correlation between non-neighbors, our algorithm successfully recovers the correct graph in such scenarios. In our simulations later we will see that the failure of the RWL algorithm for the diamond network exactly corresponds to the case $\Delta > D_{th} = \frac{2\theta}{\log\cosh(2\theta)} + 1$, which is also when the $Greedy(\epsilon)$ fails. In [16] the authors prove that for a given Δ the RWL algorithm fails when $\theta < \theta_T$ and this critical threshold θ_T behaves like $\frac{1}{\Delta}$. Now if we define

$$\theta_0 = \max\{\theta : \frac{2\theta}{\log\cosh(2\theta)} + 1 \geq \Delta\} \tag{4}$$

Then from our simulations for all $\theta < \theta_0$ the RWL algorithm fails. Also this θ_0 is almost equal to $\frac{1}{\Delta}$. Hence we make the following conjecture.

Conjecture 1: The RWL algorithm fails to recover the correct graph in the diamond network exactly when $\theta < \theta_0$. θ_0 *given by equation* (4).

In [17] Jalali et al. presented a forward-backward algorithm based on convex optimization for learning *pairwise graphical models* (as opposed to general graphical models in this paper). It has even lower sample complexity of $\Omega(\Delta^2\log p)$ and works under slightly milder assumptions than the RWL algorithm.

D. Which Greedy Algorithm Should We Use?

From the above performance comparison we can say that $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms can be used to find the graph G efficiently in discrete graphical models satisfying non-degeneracy assumption. A natural question to ask then is which among these three greedy algorithms should we use? The answer depends on the

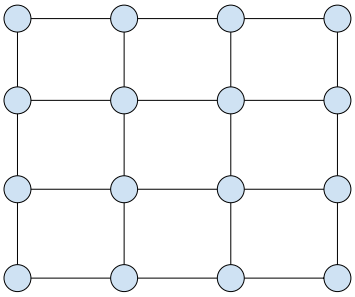

Fig. 5. A 4×4 grid with $\Delta = 4$ and $p = 16$ used for the simulation of the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms.

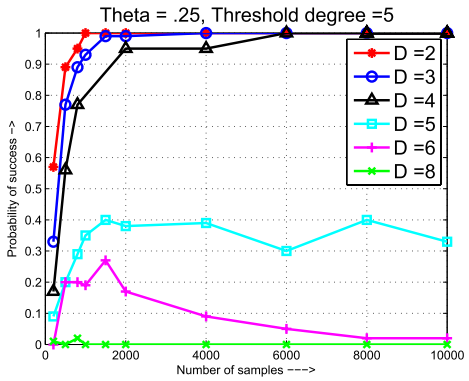

Fig. 6. Performance of the RWL algorithm in diamond network of Figure 4 for varying maximum degree with $\theta = .25$ and $D_{th} = 5$. RWL fails whenever $D > D_{th}$.

particular application. In terms of sample complexity theoretically $FbGreedy(\epsilon, \alpha)$ has a higher sample complexity than $RecGreedy(\epsilon)$ and $GreedyP(\epsilon)$ algorithms. However the difference is not much for a constant α and in our experiments we see all the three greedy algorithms have similar sample complexities (see Section VII). The theoretical guarantees on computation complexity also vary depending on parameters α and Δ. Theoretically for fixed α the $FbGreedy(\epsilon, \alpha)$ algorithm has the best expected runtime guarantee. From our experiments we see $RecGreedy(\epsilon)$ has much higher runtime than $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms which show similar run-times. We conclude that in practical applications it is better to use $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms than the $RecGreedy(\epsilon)$ algorithm. Now if we know the bound on maximum degree Δ, after running the $Greedy(\epsilon)$ algorithm if the size of the estimated neighborhood set $\widehat{N}(i)$ is considerably higher than Δ this indicates there are large number of non-neighbors. In such cases $GreedyP(\epsilon)$ may take a considerable time to remove these non-neighbors during the node pruning step and $FbGreedy(\epsilon, \alpha)$ algorithm could have removed much of these nodes in earlier iterations, when the size of the conditioning set was still small, resulting in less computation. This calls for the use of $FbGreedy(\epsilon, \alpha)$ algorithm in these cases. Similarly when size of the set $\widehat{N}(i)$ returned by the $Greedy(\epsilon)$ is comparable or slightly greater than Δ it will be more efficient to use the $GreedyP(\epsilon)$ algorithm (for example in the diamond network and grid network as shown in Section VII).

VII. Simulation Results

In this section we present some simulation results characterizing the performance of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms. We compare the performance with the $Greedy(\epsilon)$ algorithm [14] and the logistic regression based RWL algorithm [10] in Ising models. We consider two graphs, a 4×4 square grid (Figure 5) and the diamond network (Figure 4). In each case we consider an Ising model on the graphs. For the 4×4 grid we take the edge weights $\theta \in \{.25, -.25\}$, generated randomly. For the diamond network we take all equal edge weights $\theta = .25$ or .5. Independent and identically distributed samples are generated from the models using Gibbs sampling and the algorithms are run with increasing number of samples. We implement the RWL algorithm using ℓ_1-logistic regression solver by Koh et al. [20] and our algorithms using MATLAB. The parameter ϵ for the greedy algorithms and the ℓ_1 regularization parameter λ for the RWL algorithm are chosen through cross validation which gives the least estimation error on a training dataset (See Section III-D). From Theorem 1 and 2 we see that reducing α decreases the runtime of $FbGreedy(\epsilon, \alpha)$ algorithm but can increase its sample complexity. Hence for practical applications α can be chosen to trade-off between sample complexity and runtime to best suit the application requirements. In our experiments α was taken as .9.

First we show that for the diamond network (Figure 4) whenever $D > D_{th} = \frac{2\theta}{\log\cosh(2\theta)} + 1$ the RWL algorithm fails to recover the correct graph. We run the RWL algorithm in diamond network with increasing maximum degree D keeping θ fixed. We take $\theta = .25$ for which $D_{th} = \frac{2\times.25}{\log\cosh(2\times.25)} + 1 = 5.16$. The performance is shown in Figure 6. We clearly see that the failure of the RWL algorithm in diamond network corresponds exactly to the case when $D > D_{th}$. The RWL algorithm fails since it predicts a false edge between nodes 0 and $D + 1$. This is surprising since this is also the condition in Proposition 1 which describes the case when $Greedy(\epsilon)$ algorithm fails for the diamond network due to the same reason of estimating a false edge. In some sense $D = D_{th}$ marks the transition between weak and strong correlation between non-neighbors in the diamond network, and both $Greedy(\epsilon)$ and RWL algorithms fail whenever there is a strong correlation. However see next that our greedy Algorithms 2, 3 and 4 succeed even when $D > D_{th}$.

Figure 1 shows the performance of the various algorithms in the case of the diamond network with $p = 6$, $\theta = .5$ and $D = 4 > D_{th} = 3.3$. The $Greedy(\epsilon)$ and RWL algorithms are unable to recover the graph but the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ recover the true graph G, they also show the same error performance. However Figure 7 shows that $GreedyP(\epsilon)$ has a better runtime than the $RecGreedy(\epsilon)$ and $FbGreedy(\epsilon, \alpha)$ algorithms for this diamond network.

Figure 8 shows the performance of the different algorithms for a 4×4 grid network. We see that for this network the RWL algorithm shows a better sample complexity than $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ or $GreedyP(\epsilon)$ as predicted by the performance analysis. This network exhibits a

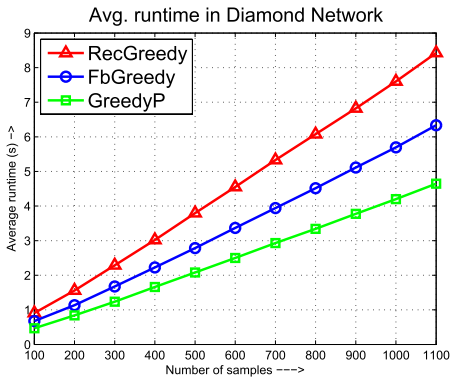

Fig. 7. Figure showing the average runtime performance of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms for the diamond network with $p = 6$, $\Delta = 4$, for varying sample size.

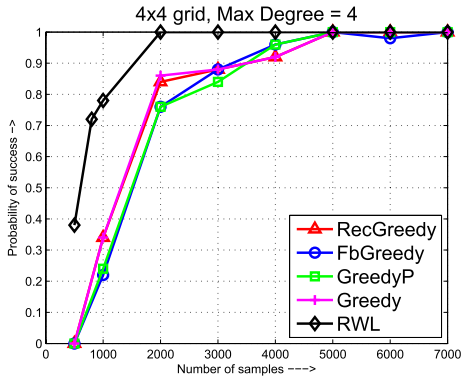

Fig. 8. Performance comparison of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$, $GreedyP(\epsilon)$, $Greedy(\epsilon)$ and RWL algorithms in a 4×4 grid with $p = 16$, $\Delta = 4$ for varying sample size. The error event is defined as $\mathcal{E} = \{\exists i \in V | \widehat{\mathcal{N}}_i \neq \mathcal{N}_i\}$. All three greedy algorithms have the same error performance for this graph.

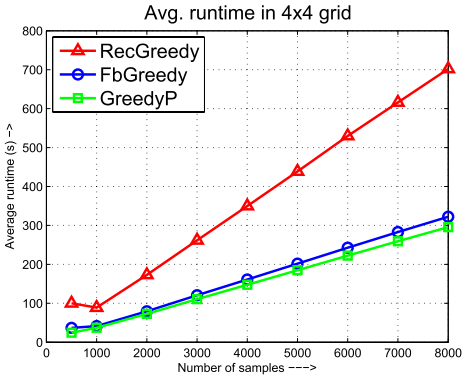

Fig. 9. Figure showing the average runtime performance of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms for the 4×4 grid network with $p = 16$, $\Delta = 4$, with varying sample size.

weak correlation among non-neighbors, hence the $Greedy(\epsilon)$ is able to correctly recover the graph, which obviously implies that the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ also correctly recovers the graph, and all have the same performance.

Figure 9 shows that the $GreedyP(\epsilon)$ algorithm also has the best runtime in the 4×4 grid network among the new greedy algorithms.

APPENDIX A
PROOF OF MAIN THEOREMS

In this section we present the proofs of Lemma 4, Theorem 1, 2 and Proposition 1.

Lemma 4:

Proof: The proof is similar to that in [14]. Let $S \subset V$ such that $|S| \leq s$. For any $i \in V$ using Azuma's inequality we get,

$$P(|\widehat{P}(x_i, x_S) - P(x_i, x_S)| > \gamma_3) \leq 2\exp(-2\gamma_3^2 n) \leq \frac{2\delta_3}{(2|\mathcal{X}|p)^{s+1}} \text{ (say)}$$

Now taking union bound over all $i \in V$, $S \subset V$, $|S| \leq s$ and all $x_i \in \mathcal{X}$, $x_S \in \mathcal{X}^{|S|}$, with probability at least $1 - \delta_3$ we have $|\widehat{P}(x_i, x_S) - P(x_i, x_S)| < \gamma_3$, for any $i \in V$, $S \subset V$, $|S| \leq s$. This implies

$$||\widehat{P}(X_i, X_S) - P(X_i, X_S)||_{TV} \leq \frac{|\mathcal{X}|^{(s+1)}}{2}\gamma_3$$
$$||\widehat{P}(X_S) - P(X_S)||_{TV} \leq \frac{|\mathcal{X}|^{(s+1)}}{2}\gamma_3$$

Now taking $\gamma_3 = \frac{\epsilon^2\alpha^2}{256|\mathcal{X}|^{s+2}}$ and using Lemma 1 we get,

$$\begin{aligned}
&|\widehat{H}(X_i|X_S) - H(X_i|X_S)| \\
&\quad \leq |\widehat{H}(X_i, X_S) - H(X_i, X_S)| + |\widehat{H}(X_S) - H(X_S)| \\
&\quad \leq |\mathcal{X}| \Bigg(\frac{2||\widehat{P}(X_i, X_S) - P(X_i, X_S)||_{TV}}{|\mathcal{X}|} \\
&\qquad \times \log \frac{|\mathcal{X}|}{2||\widehat{P}(X_i, X_S) - P(X_i, X_S)||_{TV}} \\
&\qquad + \frac{2||\widehat{P}(X_S) - P(X_S)||_{TV}}{|\mathcal{X}|} \\
&\qquad \times \log \frac{|\mathcal{X}|}{2||\widehat{P}(X_S) - P(X_S)||_{TV}} \Bigg) \\
&\quad \leq 2|\mathcal{X}|\sqrt{|\mathcal{X}|^s \gamma_3} < \frac{\alpha\epsilon}{8} = \zeta.
\end{aligned}$$

□

Theorem 1:

Proof: For this proof please refer to the detailed pseudo-code in Algorithm 5, 6 (Appendix B). The proof of correctness when $P(X)$ is known is straight forward. From local Markov property (1) the conditional entropy $H(X_i|X_{\mathcal{N}_i}) = H(X_i|X_V) < H(X_i|X_A)$ for any set A not containing all the neighbors $\mathcal{N}_i$. From degeneracy assumption (A1) including a neighboring node in the conditioning set always produce a decrease in entropy by at least ϵ. In $RecGreedy(\epsilon)$ in each iteration the algorithm runs till all the neighbors $\mathcal{N}_i$ are included in the conditioning set and the last added node is always a neighbor. In $GreedyP(\epsilon)$ nodes are added till all neighbors have been included in the conditioning set. Then in the pruning step removing a non-neighbor does not increase the entropy, therefore all spurious nodes are detected and removed. In $FbGreedy(\epsilon, \alpha)$ each iteration decrease entropy by at least $(1 - \alpha)\epsilon/2$. Since the entropy is bounded it terminates in a finite number of steps and minimum is reached only when all neighbors have been added to the conditioning set. All spurious nodes get eliminated by

the backward steps (in earlier iterations or after all neighbors are added).

Now we give the proof of sample complexity when we have samples. Define the error event $\mathcal{E} = \{\exists S \subset V, |S| < s|\ |\widehat{H}(X_i|X_S) - H(X_i|X_S)| > \frac{\epsilon}{8}\}$. Note that when $\mathcal{E}^c$ occurs we have for any $i \in V$, $j \in \mathcal{N}_i$, $A \subset V\backslash\{i, j\}$, $|A| < s$

$$\widehat{H}(X_i|X_A) - \widehat{H}(X_i|X_A, X_j) \geq H(X_i|X_A) - H(X_i|X_A, X_j) - \frac{\epsilon}{4} > \frac{3\epsilon}{4} \tag{5}$$

which follows from equation (3).

Proof for RecGreedy(ϵ) Algorithm: We first show that when $\mathcal{E}^c$ occurs the $RecGreedy(\epsilon)$ correctly estimates the graph G. The proof is by induction. Let $\mathcal{N}_i = \{j_1, \ldots, j_{\Delta_i}\} \subset V$. Let r denote the counter indicating the number of times the outermost while loop has run and s be the counter indicating the number of times the inner while loop has run in a particular iteration of the outer while loop. Clearly from Lemma 2 $s \leq \frac{2\log|\mathcal{X}|}{\epsilon}$. In the first step since $\widehat{T}(i) = \phi$ the algorithm finds the node $k \in V$ such that $\widehat{H}(X_i|X_k)$ is minimized and adds it to $\widehat{T}(i)$. Suppose it runs till $s = s_1$ such that $\mathcal{N}_i \not\subset \widehat{T}(i)$, then $\exists$ some $j_l \in \mathcal{N}_i$ such that $j_l \notin \widehat{T}(i)$. Then from equation (5) $\widehat{H}(X_i|X_{\widehat{T}(i)}, X_{j_l}) < \widehat{H}(X_i|X_{\widehat{T}(i)}) - \epsilon/2$. Hence $\min_{k\in V-\widehat{T}(i)} \widehat{H}(X_i|X_{\widehat{T}(i)}, X_k) < \widehat{H}(X_i|X_{\widehat{T}(i)}) - \epsilon/2$. Therefore the control goes to the next iteration $s = s_1 + 1$. However after the last neighbor say j_l is added to $\widehat{T}(i)$ we have

$$|\widehat{H}(X_i|X_{\widehat{T}(i)}, X_k) - \widehat{H}(X_i|X_{\widehat{T}(i)})| \leq |H(X_i|X_{\widehat{T}(i)}, X_k) - H(X_i|X_{\widehat{T}(i)})| + \frac{\epsilon}{4} = 0 + \frac{\epsilon}{4} = \frac{\epsilon}{4} < \frac{\epsilon}{2} \tag{6}$$

for any $k \in V - \widehat{T}(i)$. Thus j_l is added to $\widehat{N}_i$, variable *complete* is set to TRUE and the control exits the inner while loop going to the next iteration $r = r+1$. Proceeding similarly one neighboring node is discovered in each iteration $r = 1$ to $r = \Delta_i$. At $r = \Delta_i + 1$, $\widehat{N}(i) = \mathcal{N}_i$. Thus in step $s = 1$, $\widehat{T}(i) = \mathcal{N}_i$, so the entropy cannot be reduced further. Hence variable *iterate* is set to FALSE and control exits the outer while loop returning the correct neighborhood $\widehat{N}(i) = \mathcal{N}_i$. Lemma 2 bounds the number of steps in each iteration by $\frac{2\log|\mathcal{X}|}{\epsilon}$. Since a single node is added in each iteration the maximum size of the conditioning set is also upper bounded by $\lceil\frac{2\log|\mathcal{X}|}{\epsilon}\rceil$.

Now taking $\delta_3 = \delta$, $s = \lceil\frac{2\log|\mathcal{X}|}{\epsilon}\rceil$, $\zeta = \frac{\epsilon}{8}$ in Lemma 4 we have for $n = \Omega\left(\frac{|\mathcal{X}|^{2\log|\mathcal{X}|/\epsilon}}{\epsilon^5}\log\frac{p}{\delta}\right)$, $P(\mathcal{E}) \leq \delta$.

Therefore with probability greater than $1 - \delta$ the $RecGreedy(\epsilon)$ correctly recovers G.

Proof for FbGreedy(ϵ, α) Algorithm: Define $\mathcal{E} = \{\exists S \subset V, |S| < s|\ |\widehat{H}(X_i|X_S) - H(X_i|X_S)| > \frac{\alpha\epsilon}{8}\}$. Let s denote the number of iterations of the while loop. When $\mathcal{E}^c$ occurs we have for any $i \in V$, $j \in \mathcal{N}_i$, $A \subset V\backslash\{i, j\}$, $|A| < s$

$$\widehat{H}(X_i|X_A) - \widehat{H}(X_i|X_A, X_j) \geq H(X_i|X_A) - H(X_i|X_A, X_j) - \frac{\alpha\epsilon}{4} > \frac{3\epsilon}{4} \tag{7}$$

Again we prove by induction. For $s = 1$ the forward step adds a node to the conditioning set $\widehat{N}(i)$ as shown previously for the $RecGreedy(\epsilon)$ algorithm. Consider iteration $s > 1$. Note that it is enough to show the following.

- In each iteration the backward step never removes a neighboring node $j \in \mathcal{N}_i$.
- After the last neighbor is added to the conditioning set $\widehat{N}(i)$ the backward step removes all non-neighbors if any.

From equation (7) it is clear that removing a neighboring node $j \in \widehat{N}(i) \bigcap \mathcal{N}_i$ increases the entropy by at least $\frac{3\epsilon}{4} > \frac{\alpha\epsilon}{2}$. Hence a neighboring node is never removed in the backward step. If there exists a non-neighbor $l \in \widehat{N}(i)$ such that $\widehat{H}(X_i|X_{\widehat{N}(i)\backslash l}) - \widehat{H}(X_i|X_{\widehat{N}(i)}) < \frac{\alpha\epsilon}{2}$ and it produces the least increase in entropy then it gets removed from $\widehat{N}(i)$ and we go to iteration $s + 1$. This continues till the forward step had added all neighbors $j \in \mathcal{N}_i$ to the conditioning set. After adding the last neighbor to the conditioning set equation (6) ensures that the forward step adds no other nodes to the conditioning set $\widehat{N}(i)$. If $\widehat{N}(i) = \mathcal{N}_i$ we are done. Else for any non-neighbor $j \in \widehat{N}(i)$ we have,

$$|\widehat{H}(X_i|X_{\widehat{N}(i)\backslash j}) - \widehat{H}(X_i|X_{\widehat{N}(i)})| \leq |H(X_i|X_{\widehat{N}(i)\backslash j}) - H(X_i|X_{\widehat{N}(i)})| + \frac{\alpha\epsilon}{4} = 0 + \frac{\alpha\epsilon}{4} = \frac{\alpha\epsilon}{4} < \frac{\alpha\epsilon}{2} \tag{8}$$

Hence the backward step will remove j from the conditioning set (or any other non-neighbor that produces the least increase in entropy). This occurs till all non-neighbors are removed and $\widehat{N}(i) = \mathcal{N}_i$ when neither the forward or the backward step works. The flag complete is then set to TRUE and Algorithm 6 exits the while loop giving the correct neighborhood of node i. Again from Lemma 3 the number of steps required for convergence is bounded by $\frac{4\log|\mathcal{X}|}{(1-\alpha)\epsilon}$. At most one node is added to the conditioning set in each iteration hence the maximum size of the conditioning set is bounded by $\lceil\frac{4\log|\mathcal{X}|}{(1-\alpha)\epsilon}\rceil$. As shown previously for the $RecGreedy(\epsilon)$ algorithm from Lemma 4 with $\delta_3 = \delta$, $s = \lceil\frac{4\log|\mathcal{X}|}{(1-\alpha)\epsilon}\rceil$ and $\zeta = \frac{\alpha\epsilon}{8}$ for $n = \Omega\left(\frac{|\mathcal{X}|^{4\log|\mathcal{X}|/((1-\alpha)\epsilon)}}{(1-\alpha)\alpha^4\epsilon^5}\log\frac{p}{\delta}\right)$ the probability of error $P(\mathcal{E}) \leq \delta$. Therefore the $FbGreedy(\epsilon, \alpha)$ succeeds with probability at least $1 - \delta$. This completes the proof.

Proof for GreedyP(ϵ) Algorithm: The proof is similar to that for the $RecGreedy(\epsilon)$ algorithm. Let event $\mathcal{E}$ be as defined in the proof for $RecGreedy(\epsilon)$ algorithm. When event $\mathcal{E}^c$ occurs the $Greedy(\epsilon)$ runs till all neighbors are added to the set $\widehat{N}(i)$. Then for non-neighbors $j \in \widehat{N}(i)$

$$|\widehat{H}(X_i|X_{\widehat{N}(i)\backslash j}) - \widehat{H}(X_i|X_{\widehat{N}(i)})| \leq |H(X_i|X_{\widehat{N}(i)\backslash j}) - H(X_i|X_{\widehat{N}(i)})| + \frac{\epsilon}{4} = 0 + \frac{\epsilon}{4} = \frac{\epsilon}{4} < \frac{\epsilon}{2}$$

Hence j is removed from $\widehat{N}(i)$. But for any neighbor $k \in \mathcal{N}(i)$

$$|\widehat{H}(X_i|X_{\widehat{N}(i)\backslash k}) - \widehat{H}(X_i|X_{\widehat{N}(i)})| \geq |H(X_i|X_{\widehat{N}(i)\backslash j}) - H(X_i|X_{\widehat{N}(i)})| - \frac{\epsilon}{4} \geq \epsilon - \frac{\epsilon}{4} = \frac{3\epsilon}{4} > \frac{\epsilon}{2}$$

Thus the neighbors are not eliminated. The algorithm terminates after all non-neighbors have been eliminated. Using Lemma 4 and 2 the probability of error is upper bounded by $P(\mathcal{E}) \leq \delta$ with number of samples $n = \Omega\left(\frac{|\mathcal{X}|^{2\log|\mathcal{X}|/\epsilon}}{\epsilon^5}\log\frac{p}{\delta}\right)$. □

Theorem 2:

Proof: First consider the $RecGreedy(\epsilon)$ algorithm. With probability $1-\delta$ Algorithm 2 finds the correct neighborhood of each node i. In this case from Lemma 2 the number of steps in each recursion is $O(\frac{1}{\epsilon})$, the search in each step takes $O(p)$ time, number of recursions is at most Δ and the entropy calculation takes $O(n)$ time for each node i. Hence the overall runtime is $O(\frac{p^2}{\epsilon}\Delta n)$. When the algorithm makes an error with probability δ the number of recursions is bounded by $O(p)$. Hence the overall expected runtime is $O\left(\delta\frac{p^3}{\epsilon}n+(1-\delta)\frac{p^2}{\epsilon}\Delta n\right)$.

For the $FbGreedy(\epsilon,\alpha)$ algorithm from Lemma 3 we know that the number of steps is $O(\frac{1}{(1-\alpha)\epsilon})$. The search in either the forward or backward step is bounded by p and the entropy calculation takes $O(n)$ time. Hence when the algorithm succeeds the run time is $O(\frac{p^2}{(1-\alpha)\epsilon}n)$. Note that even when the algorithm fails with probability δ, we can prevent going into infinite loops by making sure that once the forward step stopped it is never restarted. Hence the number of steps will still be $O(\frac{1}{(1-\alpha)\epsilon})$ and the overall runtime remains the same. Thus the expected runtime is $O\left(\frac{p^2}{(1-\alpha)\epsilon}n\right)$.

In $GreedyP(\epsilon)$ algorithm when it succeeds with probability $1-\delta$, for each node $i\in V$, the $Greedy(\epsilon)$ takes at most $\frac{2\log|\mathcal{X}|}{\epsilon}$ steps. In each step search set is bounded by p and conditional entropy computation takes $O(n)$ time. After greedy algorithm terminates $|\widehat{N}(i)|\leq\frac{2\log|\mathcal{X}|}{\epsilon}$ since one node has been added in each step. Hence number iterations in pruning step is bounded by $\frac{2\log|\mathcal{X}|}{\epsilon}$ and again conditional entropy computation take $O(n)$ time. Hence the total runtime is $O\left(\frac{p^2}{\epsilon}n+\frac{p}{\epsilon}n\right)=O\left(\frac{p(p+1)}{\epsilon}n\right)$. Even when error occurs the number of greedy steps and pruning iterations is still bounded by $\frac{2\log|\mathcal{X}|}{\epsilon}$. Therefore the overall expected runtime is $O\left(\frac{p(p+1)}{\epsilon}n\right)$. □

Proposition 1:

Proof: Define $H(a)=a\log(\frac{1}{a})+(1-a)\log(\frac{1}{1-a})$ for $0\leq a\leq 1$. Then simple calculation shows $H(X_0|X_{D+1})=H(p)$ and $H(X_0|X_1)=H(q)$ where

$$p=\frac{2^{D+1}}{2^{D+1}+2(e^{2\theta}+e^{-2\theta})^D}$$
$$q=\frac{2^D+2e^{-2\theta}(e^{2\theta}+e^{-2\theta})^{D-1}}{2^{D+1}+2(e^{2\theta}+e^{-2\theta})^D}$$

Note that $p<\frac{1}{2}$ and $q<\frac{1}{2}$. Since $H(a)$ is monotonic increasing for $0<a<\frac{1}{2}$, $H(X_0|X_{D+1})<H(X_0|X_1)$ iff $p<q$. This implies,

$$\begin{aligned}2^{D+1}&<2^D+2e^{-2\theta}(e^{2\theta}+e^{-2\theta})^{D-1}\\ 2&<1+e^{-2\theta}\left(\frac{e^{2\theta}+e^{-2\theta}}{2}\right)^{D-1}\\ e^{2\theta}&<\left(\frac{e^{2\theta}+e^{-2\theta}}{2}\right)^{D-1}\\ D&>\frac{2\theta}{\log\left(\frac{e^{2\theta}+e^{-2\theta}}{2}\right)}+1=\frac{2\theta}{\log\cosh(2\theta)}+1.\end{aligned}$$

□

Algorithm 5 $RecGreedy(\epsilon)$

for $i=1$ to $|V|$ **do**
  $\widehat{N}(i)\leftarrow\phi$, iterate $\leftarrow$ TRUE
  while iterate **do**
    $\widehat{T}(i)\leftarrow\widehat{N}(i)$, last $\leftarrow 0$, complete $\leftarrow$ FALSE
    while ! complete **do**
      $j=\arg\min_{k\in V\setminus\widehat{T}(i)}\widehat{H}(X_i|X_{\widehat{T}(i)},X_k)$
      if $\widehat{H}(X_i|X_{\widehat{T}(i)},X_j)<\widehat{H}(X_i|X_{\widehat{T}(i)})-\frac{\epsilon}{2}$ **then**
        $\widehat{T}(i)\leftarrow\widehat{T}(i)\bigcup\{j\}$
        last $\leftarrow j$
      else
        if last $\neq 0$ **then**
          $\widehat{N}(i)\leftarrow\widehat{N}(i)\bigcup\{\text{last}\}$
        else
          iterate $\leftarrow$ FALSE
        end if
        complete $\leftarrow$ TRUE
      end if
    end while
  end while
end for

Algorithm 6 $FbGreedy(\epsilon,\alpha)$

for $i=1$ to $|V|$ **do**
  $\widehat{N}(i)\leftarrow\phi$, added $\leftarrow$ FALSE, complete $\leftarrow$ FALSE
  while ! complete **do** ▷ Forward Step:
    $j=\arg\min_{k\in V\setminus\widehat{N}(i)}\widehat{H}(X_i|X_{\widehat{N}(i)},X_k)$
    if $\widehat{H}(X_i|X_{\widehat{N}(i)},X_j)<\widehat{H}(X_i|X_{\widehat{N}(i)})-\frac{\epsilon}{2}$ **then**
      $\widehat{N}(i)\leftarrow\widehat{N}(i)\bigcup\{j\}$
      added $\leftarrow$ TRUE
    else
      added $\leftarrow$ FALSE
    end if ▷ Backward Step:
    $l=\arg\min_{k\in\widehat{N}(i)}\widehat{H}(X_i|X_{\widehat{N}(i)\setminus k})$
    if $\widehat{H}(X_i|X_{\widehat{N}(i)\setminus l})-\widehat{H}(X_i|X_{\widehat{N}(i)})<\frac{\alpha\epsilon}{2}$ **then**
      $\widehat{N}(i)\leftarrow\widehat{N}(i)\setminus\{l\}$
    else
      if ! added **then**
        complete $\leftarrow$ TRUE
      end if
    end if
  end while
end for

APPENDIX B
DETAILED PSEUDO-CODE

In this section we give a more detailed pseudo-code for $RecGreedy(\epsilon)$ and $FbGreedy(\epsilon,\alpha)$ algorithms useful in implementation and is used in our numerical experiments.

REFERENCES

[1] A. Ray, S. Sanghavi, and S. Shakkottai, "Greedy learning of graphical models with small girth," in *Proc. 50th Annu. Allerton Conf. Commun., Control, Comput.*, Oct. 2012, pp. 2024–2031.

[2] E. Ising, "Beitrag zur theorie des ferromagnetismus," *Zeitschrift Phys.*, vol. 31, no. 1, pp. 253–258, 1925.

[3] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA, USA: MIT Press, 1999.
[4] G. R. Cross and A. K. Jain, "Markov random field texture models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-5, no. 1, pp. 25–39, Jan. 1983.
[5] M. Hassner and J. Sklansky, "The use of Markov random fields as models of texture," *Comput. Graph. Image Process.*, vol. 12, no. 4, pp. 357–370, 1980.
[6] S. Wasserman and P. Pattison, "Logit models and logistic regressions for social networks 1. An introduction to Markov graphs and p^*," *Psychometrika*, vol. 61, no. 3, pp. 401–425, 1996.
[7] A. Dobra, C. Hans, B. Jones, J. R. Nevins, G. Yao, and M. West, "Sparse graphical models for exploring gene expression data," *J. Multivariate Anal.*, vol. 90, no. 1, pp. 196–212, 2004.
[8] D. Karger and N. Srebro, "Learning Markov networks: Maximum bounded tree-width graphs," in *Proc. 12th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2001, pp. 392–401.
[9] C. Chow and C. Liu, "Approximating discrete probability distributions with dependence trees," *IEEE Trans. Inf. Theory*, vol. 14, no. 3, pp. 462–467, May 1968.
[10] P. Ravikumar, M. J. Wainwright, and J. D. Lafferty, "High-dimensional Ising model selection using ℓ_1-regularized logistic regression," *Ann. Statist.*, vol. 38, no. 3, pp. 1287–1319, 2010.
[11] G. Bresler, E. Mossel, and A. Sly, "Reconstruction of Markov random field from samples: Some observations and algorithms," in *Proc. RANDOM*, 2008, pp. 343–356.
[12] R. Wu, R. Srikant, and J. Ni, "Learning loosely connected Markov random fields," *Stochastic Syst.*, vol. 3, no. 2, pp. 362–404, 2013.
[13] A. Jalali, P. Ravikumar, V. Vasuki, and S. Sanghavi, "On learning discrete graphical models using group-sparse regularization," in *Proc. Int. Conf. Artif. Intell. Statist. (AISTATS)*, 2011, pp. 378–387.
[14] P. Netrapalli, S. Banerjee, S. Sanghavi, and S. Shakkottai, "Greedy learning of Markov network structure," in *Proc. 48th Annu. Allerton Conf. Commun., Control, Comput.*, Sep./Oct. 2010, pp. 1295–1302.
[15] A. Anandkumar, V. Y. F. Tan, F. Huang, and A. S. Willsky, "High-dimensional structure estimation in Ising models: Local separation criterion," *Ann. Statist.*, vol. 40, no. 3, pp. 1346–1375, 2012.
[16] J. Bento and A. Montanari. (2011). "On the trade-off between complexity and correlation decay in structural learning algorithms." [Online]. Available: http://arxiv.org/abs/1110.1769
[17] A. Jalali, C. C. Johnson, and P. Ravikumar, "On learning discrete graphical models using greedy methods," in *Advances in Neural Information Processing Systems 24*. Red Hook, NY, USA: Curran Associates, 2011.
[18] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. New York, NY, USA: MIT Press, 2009.
[19] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York, NY, USA: Wiley, 2006.
[20] K. Koh, S. J. Kim, and S. Boyd. *A Largescale Solver for L1regularized Logistic Regression Problems*. http://www.stanford.edu/~boyd/l1_logreg/, accessed 2012.

Avik Ray is a Ph.D. candidate in ECE department at UT Austin. Prior to that he received his M.E. degree in Telecommunication from Indian Institute of Science, Bangalore, India in 2009 and B.E. degree in Electronics and Telecommunication Engineering from Jadavpur University, Calcutta, India in 2007. His primary research interests include Machine Learning, Networking and Wireless Communication. Avik is a recipient of MCD Fellowship from UT Austin, Prof. S. V. C. Aiya Gold Medal from Indian Institute of Science and University Gold Medal from Jadavpur University.

Sujay Sanghavi (S'02–M'06) is an Associate Professor in Electrical and Computer Engineering at the University of Texas at Austin. Sujay has an MS in ECE, and MS in Mathematics, and a PhD in ECE all from the University of Illinois, and a B. Tech in EE from IIT Bombay. Sujay's research lies in the areas of statistical inference, optimization, algorithms and networks. He has an NSF Career award, and a Young Investigator award from the DoD. He has been a visiting scientist at Google Research and Qualcomm.

Sanjay Shakkottai (M'02–SM'11–F'14) received his Ph.D. from the ECE Department at the University of Illinois at Urbana-Champaign in 2002. He is with The University of Texas at Austin, where he is currently a Professor in the Department of Electrical and Computer Engineering. He received the NSF CAREER award in 2004, and elected as an IEEE Fellow in 2014. His current research interests include network architectures, algorithms and performance analysis for wireless networks, and learning and inference over social networks.