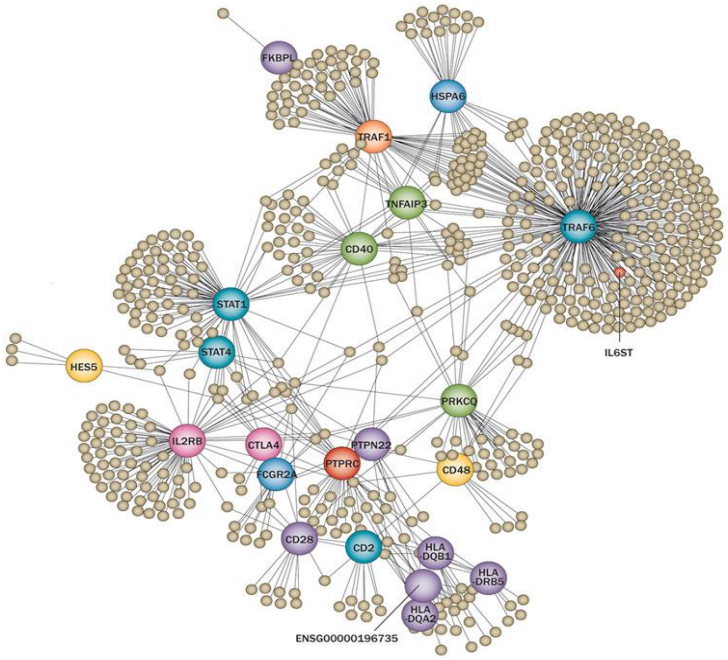
Learning Scale-Free Networks by Dynamic Node-Specific Degree Prior

Qingming Tang, Siqi Sun and **Jinbo Xu**

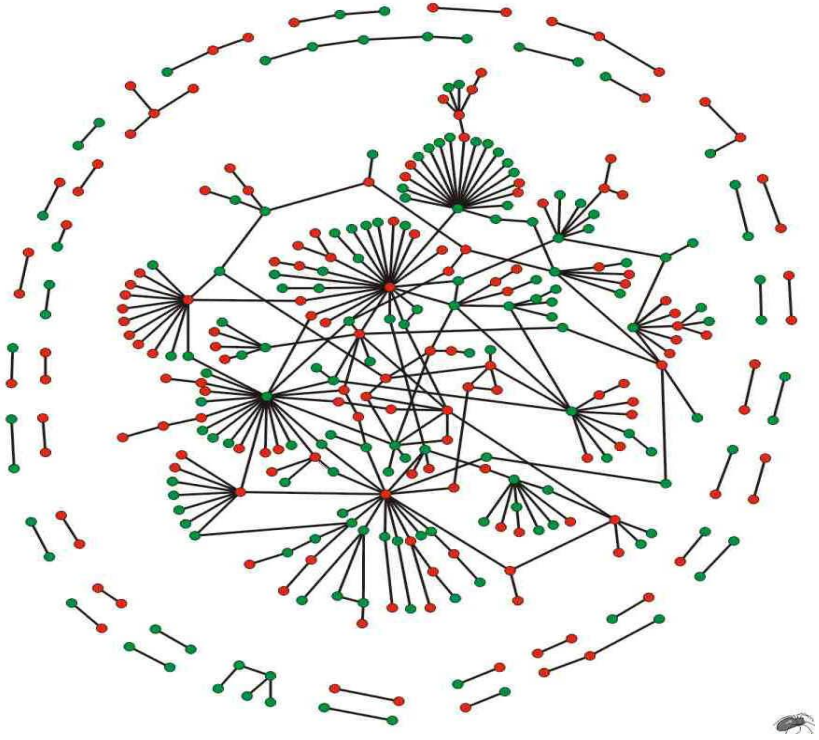
Toyota Technological Institute at Chicago

Scale-Free Networks

social network, gene expression network, protein-protein interaction network: small percentage of hub nodes, lots of nodes of small degree



Sebastien Viatte, Darren Plant & Soumya Raychaudhuri
Nature Reviews Rheumatology **9**, 141-153 (March 2013)
doi:10.1038/nrrheum.2012.23



Yeast PPI network
T. Ito, *et. al.*, PNAS 98, 4569--4574 (2001)

Learning Scale-Free Networks by Gaussian Graphical Model

Given data covariance matrix S, negative log-likelihood:

$$F(X) = tr(XS) - \log(\det(X))$$

X: precision matrix, encoding the underlying network

Objective function:

$$\min F(X) + \beta\Omega(X)$$

$\Omega(X)$: regularization

β: hyper-parameter

Power-Law Regularization

$$P(degree = k) \sim k^{-\gamma}, \gamma \in [2, 3]$$

Power-law regularizer: minimizing negative log-likelihood of degree distribution

$$\Omega(X) \sim S(G) = \sum_{i=1}^{p} \log(d(i) + 1)$$

It is hard to directly optimize this regularizer!

Approx of Power-Law Regularizer

- L_1 approx of degree (Liu et al. 2011)

$$d(i) = |X|_{-i}$$
$$= |X_{i,1}| + \cdots + |X_{i,i-1}| + |X_{i,i+1}| + \cdots + |X_{i,p}|$$

- Lovasz extension of logarithm of degree (Defazio et al. 2012)

$$\sum_{v=1}^{p} \sum_{u=1}^{p-1} |X_{v,(u)}| \Big(\log(u+1) - \log(u) \Big).$$

Larger penalty for edges more likely exist!

Where $X_{v,(u)}$ is the element with the u-th largest absolute value in $\{X_{v,1}, \ldots, X_{v,v-1}, X_{v,v+1}, \ldots, X_{v,p}\}$

Is Power-Law Regularizer S(G) Good?

Given a scale-free graph G, we can construct a non-scale-free graph G' with the same number of edges, such that $\mathrm{S(G)} > \mathrm{S(G')}$

(1) Pick two nodes u and v in G, such that $d_u = d_v = a$.

(2) Find a node x such that edge (u,x) exists, but (v,x) does not.

(3) Add (v,x) and remove (u,x) to form a new graph $G^{(1)}$.

Since $2\log(a+1) > \log(a+2) + \log(a)$, we have $\mathrm{S}\left(G^{(1)}\right) < S(G^{(0)})$

(4) Repeat this many times to generate a non-scale-free graph G'

A New Degree Prior

- Power-Law regularizer considers only global degree distribution
- We want to regularize the degree of individual node
  - Smaller penalty for nodes which may have larger degree
  - Smaller penalty for edges which more likely exist
- Challenges
  - The strength of one edge and the degree of a node is unknown.
  - If individual degree is roughly known, how to induce a graph?

Node-Specific Degree Regularizer

Given a degree distribution $\{d_1, d_2, \ldots, d_p\}$, we want a prior that favors a graph following this distribution:

$$\sum_{u=1}^{p} \frac{\sum_{j=1}^{p-1} H_j |X_{u,(j)}|}{H_{d_u}}$$

Here $H = \{H_1, H_2, \ldots, H_{p-1}\}$ is a positive moderately increasing sequence, e.g., $H_i = (\log(i+1))^{\alpha}$

Node-Specific Prior (Cont'd)

Desirable Properties

- If i< j, then $X_{u,(i)}$ has a smaller penalty than $X_{u,(j)}$
- Node u's most possible d_u edges have penalty ≤1. Others have penalty >1.
- If $d_u > d_v$, $X_{u,(i)}$ has a smaller penalty than $X_{v,(i)}$

	(1)	(2)	(3)	(4)
X_1	$1/4$	$2/4$	$3/4$	$4/4$
X_2	$1/3$	$2/3$	$3/3$	$4/4$
X_3	$1/2$	$2/2$	$3/2$	$4/2$
X_4	$1/2$	$2/2$	$3/2$	$4/2$
X_5	$1/1$	$2/2$	$3/1$	$4/1$

An example of assigned penalty to each entry in X.
Assumption:
(1) $\{X_1, X_2, X_3, X_4, X_5\}$ has degree {4,3,2,2,1}, respectively.
(2) H = {1,2,3,4}.

Node-Specific Prior (Cont'd)

Let $H \circ X_u = \sum_{j=1}^{p-1} H_j |X_{u,(j)}|$. Given two edge sets $E^{(1)}$and $E^{(2)}$ of the same size. If $E^{(1)}$ follows the degree distribution $\{d_1, d_2, \ldots, d_p\}$, then we have

$$\sum_{u=1}^{p} \frac{H \circ E_u^{(1)}}{H_{d_u}} \leq \sum_{u=1}^{p} \frac{H \circ E_u^{(2)}}{H_{d_u}}$$

This implies that our node-specific prior favors a graph following a given degree structure

Dynamic Node-Specific Prior

- By $P(k) \sim k^{-\gamma}$, estimate the number of nodes with a specific degree and then the degree of a node based upon its degree ranking.
- We can use Lovasz extension (or other methods) to rank node degree. Let τ_v be the estimated degree of a node with the "v-th" largest degree.
- Our final prior is

$$\Omega(X) = \sum_{v=1}^{p} \frac{X_{[v,X,h]} \circ H}{H_{\tau_v}}$$

- $\mathrm{h(i)} = \log(i+1) - \log(i)$, [X, h] defines a permutation of the rows in X.
- The rows of X are ranked descendingly by $X_u \circ h$, i.e., Lovasz extension of logarithm of degree
- [v,X,h] is the row with the v-th largest Lovasz extension

ADMM Algorithm

- Final formulation: *min* $F(X) + \beta\Omega(X)$

$$\Omega(X) = \sum_{v=1}^{p} \frac{X_{[v,X,h]} \circ H}{H_{\tau_v}}$$

- Introduce two variables Y and U and solve the following three steps repeatedly

$$X^{l+1} = \arg\min_X F(X) + \frac{\rho}{2}\|X - Y^l + U^l\|_F^2 \quad (11)$$

$$Y^{l+1} = \arg\min_Y \beta\Omega(Y) + \frac{\rho}{2}\|X^{l+1} - Y + U^l\|_F^2 \quad (12)$$

$$U^{l+1} = U^l + X^{l+1} - Y^{l+1} \quad (13)$$

Optimization (Node ranking update)

Let $A = X^{l+1} + U^l$ and $\lambda = \frac{\beta}{\rho}$. Minimizing (12) is equivalent to

$$\min_Y \frac{1}{2}\|Y - A\|_F^2 + \lambda\Omega(Y), \tag{14}$$

which can be relaxed as

$$\min_Y \frac{1}{2}\|Y - A\|_F^2 + \lambda \sum_{v=1}^{p} g(v) Y_{[v]} \circ H$$
$$s.t. \quad g([v]) = g([v, Y, h]) \quad 1 \le v \le p. \tag{15}$$

Where [v] is a permutation of rows in Y and [v, Y, h] is another permutation. [v, Y, h] ranks nodes (rows) by estimated node degree from high to low. Solve this problem by repeatedly updating the permutation [v].

Optimization (Cont'd)

Given a permutation [v] of rows in Y, the problem become

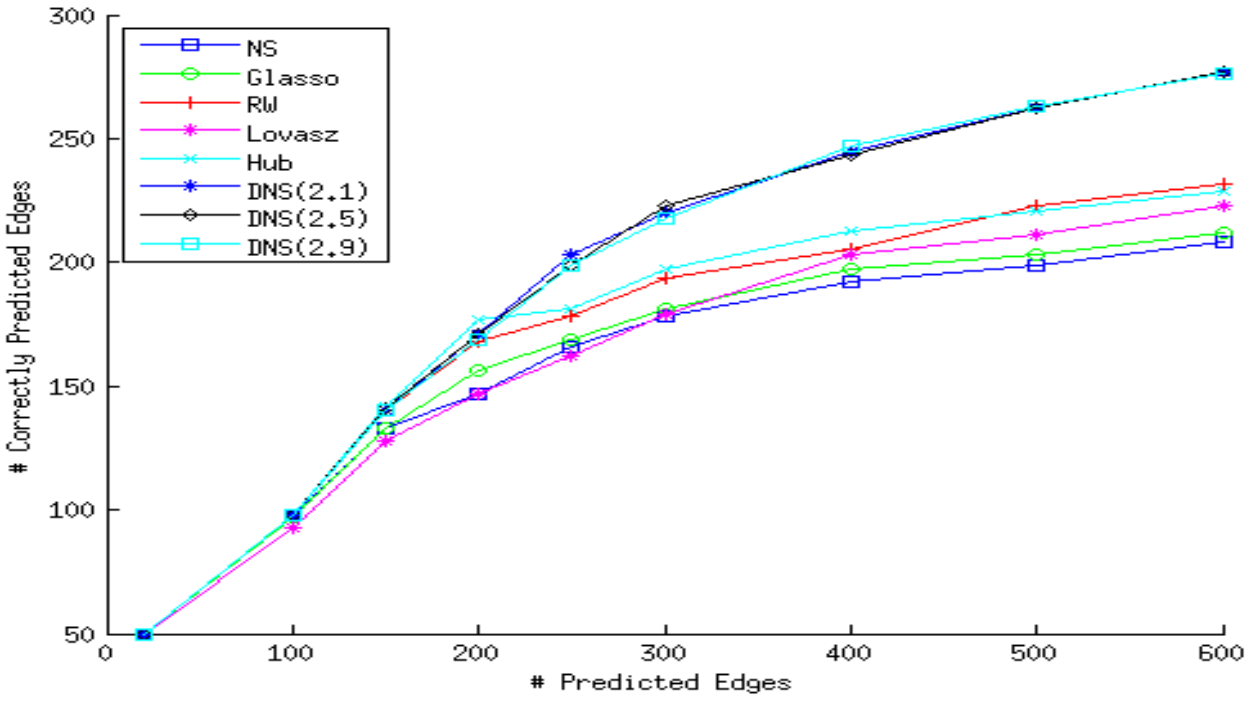
$$\min_{Y} \sum_{v=1}^{p} \{\frac{1}{2}\|Y_v - A_v\|_F^2 + Y_v \circ H'(v)\}$$
$$s.t. \quad Y = Y^T.$$

Where H'(v) is H scaled by $\lambda g(v)$.

Introducing a dual variable σ (a $p \times p$ matrix), the problem can be decomposed into p independent sub-problems as follows.

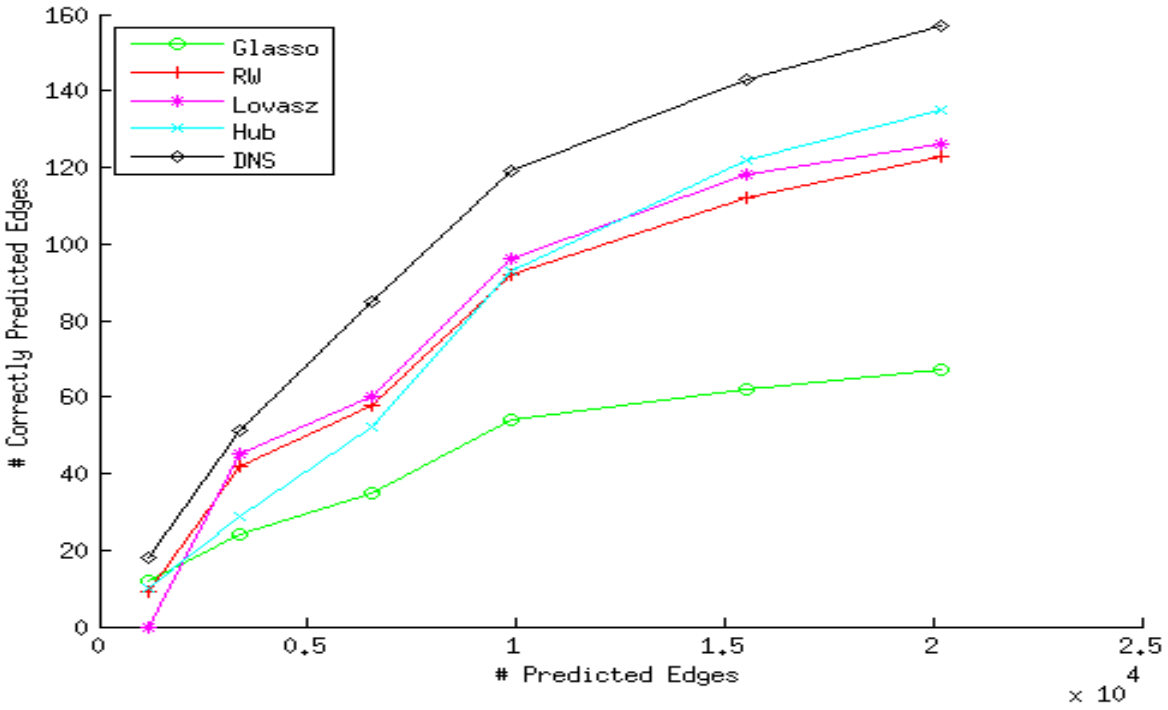
$$\min_{Y_v} \frac{1}{2}\|Y_v - (A + \sigma^T - \sigma)_v\|_F^2 + Y_v \circ H'(v)$$

Result on Simulated Data



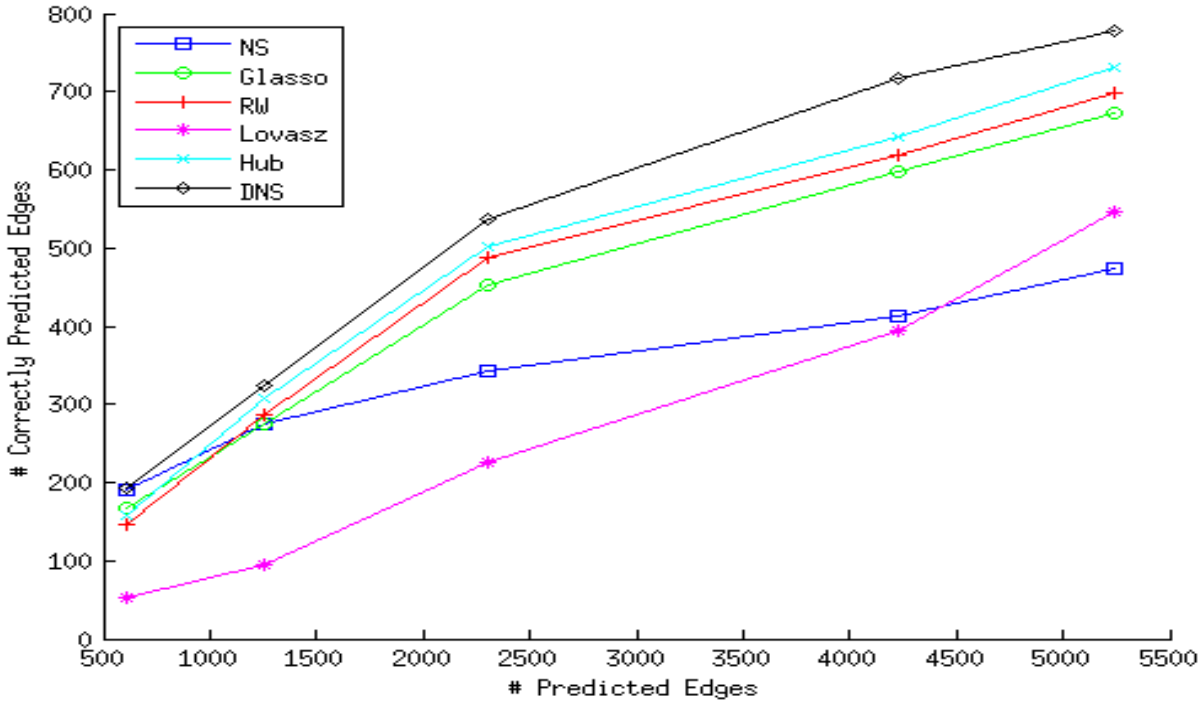
- n = 250 samples and p = 500 nodes.
- X-axis is #predicted edges and Y-axis #correctly predicted edges.
- Three different power-law distributions tested: $\gamma = 2.1, 2.5, 2.9$.

Result on Real Data



- A real gene expression data set (DREAM5 dataset 3).
- 805 samples, 4511 genes and 3902 true edges.

Result on Hub Networks



- n = 806 samples and p = 1643 nodes.
- The first data set in DREAM5 challenge. The ground truth network consists of several hubs, but the degree does not strictly follow the power-law distribution

Conclusion

- ✓ Regulation for each node and potential edge performs better than regulation for global degree distribution (i.e., the power-law regularizer)
- ✓ If we know node degree ranking, power-law distribution allows us to estimate node degree very well

Acknowledgements

- The software available at https://bitbucket.org/qmtang/scale_free/
- Students: Qingming Tang, Siqi Sun
- Funding
  - NSF CAREER award
- Computational resources
  - University of Chicago RCC

Exponent of Power-Law Distribution

- One question is we actually do not know the value of γ. And we leave γ as a hyper parameter in our model.
- Typically, by increasing γ, we would reduce the number of “hubs”. Meanwhile, the degree of the induced hubs would increase.
- By decreasing γ, we are trying to make the edges distribute more “uniformly”. When no prior knowledge is given, using a relatively smaller γ is preferable.
- In the paper, we do all the experiments using γ=2.5

Dynamic Node-Specific Prior

- By $P(k) \sim k^{-\gamma}$, we can estimate the number of nodes with a specific degree and then the degree of a node based upon its degree ranking.
- Let τ_k be the estimated degree of a node with the "k-th" largest degree. We use a ranking-based prior:

$$\Omega(X) = \sum_{v=1}^{p} \frac{X_{[v]} \circ H}{H_{\tau_v}}$$

$$X_v \circ H = \sum_{j=1}^{p-1} H_j |X_{v,(j)}|$$

[v] refers to the node with the v-th largest degree

Optimization (Edge Update)

- By introducing a *p*×*p* matrix σ, which would be updated as $\sigma = \sigma + \mu(Y - Y^T)$ during dual decomposition, the problem can be decomposed into p independent sub-problems as follows, in each iteration of dual decomposition

$$\min_{Y_v} \frac{1}{2}\|Y_v - (A + \sigma^T - \sigma)_v\|_F^2 + Y_v \circ H'(v)$$

- For simplicity, we denote $B = A + \sigma^T - \sigma$

Optimization (Edge Update)

Solving $\min_{Y_v}\{\frac{1}{2}|Y_v - B_v|_F{}^2 + Y_v \circ H'(v)\}$ is equivalent to finding an optimal partition of sorted $|B_v|$ where $|B_v| = \{|B_{v,1}|, \dots, |B_{v,p}|\}$

Thus, there exists $O(plog(p))$ algorithm for solving the above optimization problem. See our supplementary for details.

Experiment (Setting)

- Simulated scale-free network is generated by BA model (Barabási–Albert model, a famous model to generate scale-free network)
- We generate a network with 500 nodes. Each entry Ω_{uv} is set to 0.3 if uv forms an edge and 0 otherwise. The final precision Ω is obtained by setting the diagonal elements of Ω to the maximum eigenvalues of current Ω plus 0.2. We generate 250 Gaussian Samples using this final Ω.
- We also tested our method using well-known Dream5 dataset. There are four networks. The first network is simulated gene expression data, whose ground truth network contains a few hubs, but the degree distribution does not strictly follows the power law distribution.
- The other three networks are real gene expression data with known labels. The three real data sets are with significant level of noise and are hard to analyze. The performance of all the teams competing at Dream5 contest are not good.

Summary and Future Work

- Take-home messages
  - A dynamic node-specific degree prior performs better than the power-law regularizer in promote a scale-free network
  - Power-law distribution allows us to estimate node degree very well, coupled with node degree ranking
  - Dual decomposition to solve the resultant optimization problem
- Future Work
  - Automatically determining γ (the parameter for power-law distribution). However, γ does not impact the result very much as long as it is in [2,3]
  - Speed up to solve very large scale problem
  - Apply this idea to link prediction by matrix factorization