Limits of Learning in Incomplete Networks

Timothy LaRock
larock.t@husky.neu.edu

In collaboration with

Timothy Sakharov
Sahely Bhadra
Tina Eliassi-Rad

Supported by NSF CNS-1314603 & NSF IIS-1741197
Background: Incomplete Networks

- Network data is often incomplete
- Acquiring more data is often expensive and/or hard
- Research question: Given a networked dataset and limited resources to collect more data, how can you get the most bang for your buck?
Two general approaches to network completion

Don’t collect more data

Collect more data
Two general approaches to network completion

**Don’t collect more data**

- Assume a network model
- Combine network model with incomplete data to get a model of the network structure
- Infer missing data from this model
  - [Kim et al. 2011]
  - [Chen et al. 2018]

**Collect more data**
Two general approaches to network completion

**Don’t collect more data**

Assume a network model

Combine network model with incomplete data to get a model of the network structure

Infer missing data from this model

- [Kim et al. 2011]
- [Chen et al. 2018]

**Collect more data**

Estimate Statistics from partially observed network

- [Soundarajan et al. 2015]
- [Soundarajan et al. 2016]

Utilize an explore-exploit approach

- [Pfeiffer III et al. 2014]
- [Soundarajan et al. 2017]
- [Murai et al. 2018]
- [Madhawa et al. 2018]
- This work!
Our solution: Network Online Learning (NOL)

- Research Question: Given a networked dataset and limited resources to collect more data, how can you get the most bang for your buck?

- Learn to grow an incomplete network through sequential, optimal queries (to some API)
- Agnostic to both *data distributions* and *sampling method*
- Interpretable features that are computable online
Assumptions

We assume...

We know the API access model (complete vs incomplete queries).

The underlying network is static (probing the same node twice gives no new information).

We do not assume...

A model of the underlying graph.

How the initial sample was collected.
Network Online Learning (NOL)

Inputs:
- $\hat{G}_0$: Incomplete network
- $b$: probing budget
- $r$: reward function

Output:
Network after $b$ probes
$\hat{G}_b \approx G$

Example reward function:
- number of new nodes observed
NOL algorithm

**Input**: \( \hat{G}_0 \) (initial incomplete network), \( b \) (probing budget), \( p \) (jump rate), and \( \alpha \) (step-size for gradient descent)

**Output**: \( \theta \) (parameters of the learning model), \( \hat{G}_b \) (network after \( b \) probes)

1. **Initialize**: \( \theta_0 \) (randomly or heuristically); \( P_0 = \emptyset \)
2. **repeat**
3. \( \phi_t(i), \forall i \in \hat{V}_t - P_t \) \{Calculate feature vectors\}
4. \( V_{\theta_t}(\phi_t(i)) = \theta_t^T \phi_t(i) \) \{Calculate estimated rewards\}
5. With probability \( p \), choose node \( u_t \in \hat{G}_0 - P_t \) uniformly at random. \{Explore\}
6. With probability \( 1 - p \), \( u_t = \text{argmax}_i \theta_t^T \phi_t(i) \), where \( i \in \hat{V}_t - P_t \) \{Exploit\}
7. **Probe** node \( u_t \)
8. Update the observed graph \( \hat{G}_{t+1} = \{\hat{G}_t \cup \text{neighbors of } u_t\} \)
9. Collect reward \( r_t = |\hat{G}_{t+1}| - |\hat{G}_t| \)
10. Online loss \( \ell_t = (r_t - V_{\theta_t}(\phi_t(u_t)))^2 \)
11. Compute on-line gradient \( \nabla_{\theta_t} \ell_t = -2 (r_t - V_{\theta_t}(\phi(u_t))) \phi_t(u_t) \)
12. Update parameters \( \theta_{t+1} = \theta_t + \alpha \nabla_{\theta_t} \ell_t \)
13. Normalize parameters \( \theta_{t+1} = \frac{\theta_{t+1}}{||\theta_{t+1}||_2} \)
14. \( t \leftarrow t + 1 \)
15. **until** \( t = b \)
16. **return** \( \theta_b \) and \( \hat{G}_b \)
NOL algorithm

**Input:** $\hat{G}_0$ (initial incomplete network), $b$ (probing budget), $p$ (jump rate), and $\alpha$ (step-size for gradient descent)

**Output:** $\theta$ (parameters of the learning model), $\hat{G}_b$ (network after $b$ probes)

1: Initialize $\theta_0$ (randomly or heuristically); $P_0 = \emptyset$

2: repeat

3: $\phi_t(i), \forall$ node $i \in \hat{V}_t - P_t$ (Calculate feature vectors)

4: $\mathcal{V}_{\theta_t}(\phi_t(i)) = \theta^T_t \phi_t(i)$ (Calculate estimated rewards)

5: With probability $p$, choose node $u_t \in \hat{G}_0 - P_t$ uniformly at random. (Explore)

6: With probability $1 - p$, $u_t = \arg\max_i \theta^T_t \phi_t(i)$, where $i \in \hat{V}_t - P_t$ (Exploit)

7: Probe node $u_t$

8: Update the observed graph $\hat{G}_{t+1} = \{\hat{G}_t \cup$ neighbors of $u_t\}$

9: Collect reward $r_t = |\hat{G}_{t+1}| - |\hat{G}_t|$

10: Online loss $t = (r_t - \mathcal{V}_{\theta_t}(\phi_t(u_t)))^2$

11: Compute on-line gradient $\nabla_{\theta_t} loss_t = -2 (r_t - \mathcal{V}_{\theta_t}(\phi(u_t))) \phi_t(u_t)$

12: Update parameters $\theta_{t+1} = \theta_t + \alpha \nabla_{\theta_t} loss_t$

13: Normalize parameters $\hat{\theta}_{t+1} = \frac{\theta_{t+1}}{\|\theta_{t+1}\|_2}$

14: $t \leftarrow t + 1$

15: until $t = b$

16: return $\theta_b$ and $\hat{G}_b$
NOL algorithm

```
Input: \( \hat{G}_0 \) (initial incomplete network), \( b \) (probing budget), \( \rho \) (jump rate), and \( \alpha \) (step-size for gradient descent)
Output: \( \theta \) (parameters of the learning model), \( \hat{G}_b \) (network after \( b \) probes)
1: Initialize: \( \theta_0 \) (randomly or heuristically); \( P_0 = \emptyset \)
2: repeat
3: \( \phi_t(i), \forall \text{ node } i \in \hat{V}_t - P_t \) \{Calculate feature vectors\}
\( \mathcal{V}_{\theta_t}(\phi_t(i)) = \theta_t^T \phi_t(i) \) \{Calculate estimated rewards\}
4: With probability \( \rho \), choose node \( u_t \in \hat{G}_0 - P_t \) uniformly at random. \{Explore\}
5: With probability \( 1 - \rho \), \( u_t = \text{argmax}_i \theta_t^T \phi_t(i) \), where \( i \in \hat{V}_t - P_t \) \{Exploit\}
6: Probe node \( u_t \)
7: Update the observed graph \( \hat{G}_{t+1} = \{\hat{G}_t \cup \text{neighbors of } u_t\} \)
8: Collect reward \( r_t = |\hat{G}_{t+1}| - |\hat{G}_t| \)
9: Online loss \( t = (r_t - \mathcal{V}_{\theta_t}(\phi_t(u_t)))^2 \)
10: Compute on-line gradient \( \nabla_{\theta_t} \text{loss}_t = -2 (r_t - \mathcal{V}_{\theta_t}(\phi(u_t))) \phi_t(u_t) \)
11: Update parameters \( \theta_{t+1} = \theta_t + \alpha \nabla_{\theta_t} \text{loss}_t \)
12: Normalize parameters \( \theta_{t+1} = \frac{\theta_{t+1}}{||\theta_{t+1}||_2} \)
13: \( t \leftarrow t + 1 \)
14: until \( t = b \)
15: return \( \theta_b \) and \( \hat{G}_b \)
```
NOL algorithm

1. Observe the current state
   \( \phi_t(i), \forall \text{ node } i \in \hat{V}_t - P_t \) {Calculate feature vectors}
   \( V_{\theta_t}(\phi_t(i)) = \theta_t^T \phi_t(i) \) {Calculate estimated rewards}

2. With probability \( p \), choose node \( u_t \in \hat{G}_0 - P_t \) uniformly at random. {Explore}

3. With probability \( 1 - p \), \( u_t = \arg\max_i \theta_t^T \phi_t(i) \), where \( i \in \hat{V}_t - P_t \) {Exploit}

4. Update the observed graph \( \hat{G}_{t+1} \) by adding the edge \( u_t \) to \( \hat{G}_t \)

5. Collect reward \( r_t = |\hat{G}_{t+1}| - |\hat{G}_t| \)

6. Online loss \( l_t = (r_t - V_{\theta_t}(\phi_t(u_t)))^2 \)

7. Compute on-line gradient \( \nabla_{\theta_t} l_t = -2 (r_t - V_{\theta_t}(\phi(u_t))) \phi_t(u_t) \)

8. Update parameters \( \theta_{t+1} = \theta_t + \alpha \nabla_{\theta_t} l_t \)

9. Normalize parameters \( \theta_{t+1} = \frac{\theta_{t+1}}{||\theta_{t+1}||_2} \)

10. \( t \leftarrow t + 1 \)

11. until \( t = b \)

12. return \( \theta_b \) and \( \hat{G}_b \)
NOL algorithm

Observe the current state

Input: $\hat{G}_0$ (initial incomplete network), $b$ (probing budget), $p$ (jump rate), and $\alpha$ (step-size for gradient descent)
Output: $\theta$ (parameters of the learning model), $\hat{G}_b$ (network after $b$ probes)

1. Initialize: $\theta_0$ (randomly or heuristically); $P_0 = \emptyset$
2. repeat
   3. $\phi_t(i), \forall \text{ node } i \in \hat{V}_t - P_t$ \{Calculate feature vectors\}
      $V_{\theta_t}(\phi_t(i)) = \theta_T^t \phi_t(i)$ \{Calculate estimated rewards\}
   4. With probability $p$, choose node $u_t \in \hat{G}_0 - P_t$ uniformly at random. \{Explore\}
   5. With probability $1 - p$, $u_t = \arg \max_i \theta_T^t \phi_t(i)$, where $i \in \hat{V}_t - P_t$ \{Exploit\}
   6. Probe node $u_t$
   7. Update the observed graph $\hat{G}_{t+1} = \{\hat{G}_t \cup \text{ neighbors of } u_t\}$
   8. Collect reward $r_t = |\hat{G}_{t+1}| - |\hat{G}_t|$
   9. Online loss $t = (r_t - V_{\theta_t}(\phi_t(u_t)))^2$
   10. Compute on-line gradient $\nabla_{\theta_t} loss_t = -2 (r_t - V_{\theta_t}(\phi(u_t))) \phi_t(u_t)$
   11. Update parameters $\theta_{t+1} = \theta_t + \alpha \nabla_{\theta_t} loss_t$
   12. Normalize parameters $\theta_{t+1} = \frac{\theta_{t+1}}{||\theta_{t+1}||_2}$
   13. $t \leftarrow t + 1$
   14. until $t = b$
15. return $\theta_b$ and $\hat{G}_b$
NOL algorithm

Input: $\hat{G}_0$ (initial incomplete network), $b$ (probing budget), $p$ (jump rate), and $\alpha$ (step-size for gradient descent)
Output: $\theta$ (parameters of the learning model), $\hat{G}_b$ (network after $b$ probes)

1: Initialize: $\theta_0$ (randomly or heuristically); $P_0 = \emptyset$
2: repeat
3: $\phi_t(i)$, $\forall$ node $i \in \hat{V}_t - P_t$ \{Calculate feature vectors\}
4: $\nabla_{\theta_t} \phi_t(i) = \theta_t^T \phi_t(i)$ \{Calculate estimated rewards\}

With probability $p$, choose node $u_t \in G_0 - P_t$ uniformly at random. \{Explore\}
With probability $1 - p$, $u_t = \text{argmax}_i \theta_t^T \phi_t(i)$, where $i \in \hat{V}_t - P_t$ \{Exploit\}

3: Probe node $u_t$
4: Update the observed graph $\hat{G}_{t+1} = \{\hat{G}_t \cup \text{neighbors of } u_t\}$
5: Collect reward $r_t = |\hat{G}_{t+1}| - |\hat{G}_t|$
6: Online loss $t = (r_t - \nabla_{\theta_t} \phi_t(u_t))^2$
7: Compute on-line gradient $\nabla_{\theta_t} \text{loss}_t = -2 (r_t - \nabla_{\theta_t} \phi_t(u_t)) \phi_t(u_t)$
8: Update parameters $\theta_{t+1} = \theta_t + \alpha \nabla_{\theta_t} \text{loss}_t$
9: Normalize parameters $\theta_{t+1} = \frac{\theta_{t+1}}{\|\theta_{t+1}\|_2}$
10: $t \leftarrow t + 1$
11: until $t = b$
12: return $\theta_b$ and $\hat{G}_b$
NOL algorithm

1: Observe the current state
2: Choose the next action
3: Take action, update network, collect reward

Input: $\hat{G}_0$ (initial incomplete network), $b$ (probing budget), $p$ (jump rate), and $\alpha$ (step-size for gradient descent)
Output: $\theta$ (parameters of the learning model), $\hat{G}_b$ (network after $b$ probes)

1: Initialize: $\theta_0$ (randomly or heuristically); $P_0 = \emptyset$
2: repeat
3: $\phi_t(i), \forall \text{ node } i \in \hat{V}_t - P_t$ \{Calculate feature vectors\}
4: $V_{\theta_t}(\phi_t(i)) = \theta_T^T \phi_t(i)$ \{Calculate estimated rewards\}
5: With probability $p$, choose node $u_t \in \hat{G}_0 - P_t$ uniformly at random. \{Explore\}
6: With probability $1 - p$, $u_t = \arg\max_{i} \theta_T^T \phi_t(i)$, where $i \in \hat{V}_t - P_t$ \{Exploit\}
7: Probe node $u_t$
8: Update the observed graph $\hat{G}_{t+1} = \{\hat{G}_t \cup \text{ neighbors of } u_t\}$
9: Collect reward $r_t = |\hat{G}_{t+1}| - |\hat{G}_t|$
10: Online loss $t = (r_t - V_{\theta_t}(\phi_t(u_t)))^2$
11: Compute on-line gradient $\nabla_{\theta_t} \text{loss}_t = -2(r_t - V_{\theta_t}(\phi(u_t))) \phi_t(u_t)$
12: Update parameters $\theta_{t+1} = \theta_t + \alpha \nabla_{\theta_t} \text{loss}_t$
13: Normalize parameters $\theta_{t+1} = \frac{\theta_{t+1}}{||\theta_{t+1}||_2}$
14: $t \leftarrow t + 1$
15: until $t = b$
16: return $\theta_b$ and $\hat{G}_b$
NOL algorithm

1. Initialize: $\theta_0$ (randomly or heuristically); $P_0 = \emptyset$
2. repeat
   3. $\phi_t(i), \forall$ node $i \in \hat{V}_t - P_t$ \{Calculate feature vectors\}
   4. $V_{\theta_t}(\phi_t(i)) = \theta_T^T \phi_t(i)$ \{Calculate estimated rewards\}
   5. With probability $p$, choose node $u_t \in \hat{G}_0 - P_t$ uniformly at random. \{Explore\}
   6. With probability $1 - p$, $u_t = \text{argmax}_i \theta_T^T \phi_t(i)$, where $i \in \hat{V}_t - P_t$ \{Exploit\}
   7. Probe node $u_t$
   8. Update the observed graph $\hat{G}_{t+1} = \{\hat{G}_t \cup \text{neighbors of } u_t\}$
   9. Collect reward $r_t = |\hat{G}_{t+1}| - |\hat{G}_t|$
   10. Online loss $l_t = (r_t - V_{\theta_t}(\phi_t(u_t)))^2$
   11. Compute on-line gradient $\nabla_{\theta_t} l_t = -2 (r_t - V_{\theta_t}(\phi(u_t))) \phi_t(u_t)$
   12. Update parameters $\theta_{t+1} = \theta_t + \alpha \nabla_{\theta_t} l_t$
   13. Normalize parameters $\theta_{t+1} = \frac{\theta_{t+1}}{\|\theta_{t+1}\|_2}$
   14. $t \leftarrow t + 1$
3. until $t = b$
4. return $\theta_b$ and $\hat{G}_b$
**NOL algorithm**

```plaintext
Observe the current state

Choose the next action

Take action, update network, collect reward

Update parameters

---

**Online linear regression following Strehl et al., NIPS 2008.**

```
NOL algorithm

**Inputs:**
- \( G_0 \) (initial incomplete network), \( b \) (probing budget), \( p \) (jump rate), \( \alpha \) (step-size for gradient descent)

**Output:**
- \( \theta \) (parameters of the learning model), \( \hat{G}_b \) (network after \( b \) probes)

1. Initialize: \( \theta_0 \) (randomly or heuristically); \( P_0 = \emptyset \)
2. **repeat**
   3. \( \phi_t(i), \forall \text{ node } i \in \hat{V}_t - P_t \) \{Calculate feature vectors\}
   4. \( V_{\theta_t}(\phi_t(i)) = \theta_t^T \phi_t(i) \) \{Calculate estimated rewards\}
   5. With probability \( p \), choose node \( u_t \in \hat{G}_0 - P_t \) uniformly at random. \{Explore\}
   6. With probability \( 1-p \), \( u_t = \text{argmax}_i \theta_t^T \phi_t(i) \), where \( i \in \hat{V}_t - P_t \) \{Exploit\}
   7. Probe node \( u_t \)
   8. Update the observed graph \( \hat{G}_{t+1} = \{\hat{G}_t \cup \text{neighbors of } u_t\} \)
   9. Collect reward \( r_t = |\hat{G}_{t+1}| - |\hat{G}_t| \)
   10. Online loss \( l_t = (r_t - V_{\theta_t}(\phi_t(u_t)))^2 \)
   11. Compute online gradient \( \nabla_{\theta_t} l_t = -2 (r_t - V_{\theta_t}(\phi(u_t))) \phi_t(u_t) \)
   12. Update parameters \( \theta_{t+1} = \theta_t + \alpha \nabla_{\theta_t} l_t \)
   13. Normalize parameters \( \theta_{t+1} = \frac{\theta_{t+1}}{\|\theta_{t+1}\|_2} \)
   14. \( t \leftarrow t + 1 \)
3. **until** \( t = b \)
4. **return** \( \theta_b \) and \( \hat{G}_b \)
NOL algorithm

1. Observe the current state
2. Choose the next action
3. Take action, update network, collect reward
4. Update parameters
5. Repeat until budget depleted
Features
Features
Feature: In-sample degree

\[ \phi_{i,0} = \hat{d}_i \]
Feature: In-sample clustering coefficient

$$\phi_{i,1} = \hat{C}_i$$
Feature: Normalized size of connected component

\[ \phi_{i,2} = \text{CompSize}_i \]
Feature: Fraction of probed neighbors

$$\phi_{i,3} = \text{ProbedNeighbors}_i$$
Feature: Lost Reward

$\phi_{i,4} = \text{LostReward}_i$

Key idea: The order in which we probe nodes can impact the reward they yield.
Feature: Lost Reward

\[ \phi_{i,4} = \text{LostReward}_i \]

Key idea: The order in which we probe nodes can impact the reward they yield.
Research Question: Given a networked dataset, and limited resources to collect more data, how can you get the most bang for your buck?

- Potential bang for your buck depends on network structure!
Heterogeneity in network properties creates a learning “spectrum”
Heterogeneity in network properties creates a learning “spectrum”
Heterogeneity in network properties creates a learning “spectrum”

<table>
<thead>
<tr>
<th>Learning not useful</th>
<th>Potential for learning</th>
<th>Heuristics optimal</th>
</tr>
</thead>
</table>

Northeastern University
Network Science Institute
Heterogeneity in network properties creates a learning “spectrum”

Learning not useful  Potential for learning  Heuristics optimal

Homogeneous degree dist.
Heterogeneity in network properties creates a learning “spectrum”

Learning not useful

Potential for learning

Heuristics optimal\(^1\)

Homogeneous degree dist.

Heterogenous degree dist.

\(^1\)Avrachenkov et al. *INFOCOM WKSHP* (2014)
Heterogeneity in network properties creates a learning “spectrum”

Learning not useful  
Homogeneous degree dist.  

Potential for learning  
Heterogenous degree dist. with rich structure

Heuristics optimal\(^1\)  
Heterogenous degree dist.

\(^1\) Avrachenkov et al. *INFOCOM WKSHPS* (2014)
Experiments
Heuristic Baselines

- **High degree**
  - Probe the unprobed node with maximum degree
- **High degree w/ jump**
  - Probe the unprobed node with maximum degree, randomly jump with probability $p$
- **Low degree**
  - Probe the unprobed node with minimum degree
- **Random**
  - Probe a node chosen uniformly at random from the unprobed nodes
iKNN-UCB [Madhawa+ ArXiv preprint, 2018]

- K-Nearest Neighbors Upper Confidence Bound
  - Nonparametric multi-armed bandit approach
- Choose node to probe by combining nearest neighbor reward information (based on Euclidean distance between feature vectors) + extent of previous exploration of similar actions
Heterogeneity in network properties creates a learning “spectrum”

- Homogeneous degree dist.
- Potential for learning
- Heterogenous degree dist. with rich structure
- Heuristics optimal
- Learning not useful
Heterogeneity in network properties creates a learning “spectrum”

- Learning not useful
- Homogenous degree dist.
- Heuristics optimal
- Heterogenous degree dist.
- Potential for learning
- Heterogenous degree dist. with rich structure
Heterogeneity in network properties creates a learning “spectrum”

Learning not useful
Erdos-Renyi Model

Potential for learning
Heterogenous degree dist. with rich structure

Heuristics optimal
Heterogenous degree dist.
Heterogeneity in network properties creates a learning “spectrum”

- Learning not useful
- Potential for learning
- Heuristics Optimal

Erdos-Renyi Model

BTER Model\(^1\)

Heterogenous degree dist.

Heterogeneity in network properties creates a learning “spectrum”

- Learning not useful
  - Erdős-Rényi Model

- Potential for learning
  - BTER Model

- Heuristics Optimal
  - Barabási-Albert Model

Results - Learning Spectrum

- Learning not useful
- Potential for learning
- Heuristics Optimal

Erdos-Renyi Model
BTER Model
Barabasi-Albert Model

Graph showing average cumulative reward vs. number of probes with different models and parameters.
Results - Learning Spectrum

Learning not useful

Erdos-Renyi Model

Potential for learning

BTER Model

Heuristics Optimal

Barabasi-Albert Model

All methods are indistinguishable, learning doesn’t help or hurt!
Results - Learning Spectrum

Learning not useful

Potential for learning

Heuristics Optimal

Erdos-Renyi Model

BTER Model

Barabasi-Albert Model

All methods are indistinguishable, learning doesn’t help or hurt!
Results - Learning Spectrum

Learning not useful

Potential for learning

Heuristics Optimal

Erdos-Renyi Model

BTER Model

Barabasi-Albert Model

All methods are indistinguishable, learning doesn’t help or hurt!

High degree heuristic is optimal in BA model, but NOL learns to probe like the heuristic!
Results - Learning Spectrum

Learning not useful

Erdos-Renyi Model

All methods are indistinguishable, learning doesn’t help or hurt!

BTER Model

High degree heuristic is optimal in BA model, but NOL learns to probe like the heuristic!
Heterogeneity in network properties creates a learning “spectrum”

<table>
<thead>
<tr>
<th>Learning Not Useful</th>
<th>Potential for learning</th>
<th>Heuristics Optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBLP</td>
<td>Cora</td>
<td>Enron</td>
</tr>
<tr>
<td>Coauthorship network</td>
<td>Citation network</td>
<td>Email Communication Network</td>
</tr>
<tr>
<td>$N = 6.7k$</td>
<td>$N = 23k$</td>
<td>$N = 36.7k$</td>
</tr>
<tr>
<td>$E = 17k$</td>
<td>$E = 89k$</td>
<td>$E = 184k$</td>
</tr>
<tr>
<td>$\Delta s = 21.6k$</td>
<td>$\Delta s = 78.7k$</td>
<td>$\Delta s = 727k$</td>
</tr>
</tbody>
</table>
Heterogeneity in network properties creates a learning “spectrum”

Learning Not Useful

Potential for learning

Heuristics Optimal

DBLP

Cora

Enron

Average Cumulative Reward

Number of Probes

Average Cumulative Reward

Number of Probes

Average Cumulative Reward

Number of Probes

Northeastern University
Network Science Institute
Summary

- **Network Online Learning** can learn to probe online with minimal assumptions

- Success is tied to properties of the underlying network:
  - Spectrum based on objective function being maximized (degree distribution in these experiments)
  - NOL can learn to behave like the optimal heuristic

- Preliminary experiments suggest some real world complex networks fall in the “learnable” category
Thanks!

Tim LaRock
larock.t@husky.neu.edu

References


Avrachenkov et al. (2014). Pay few, influence most: Online myopic network covering. *Proceedings - IEEE INFOCOM.*


