Graph Detection Theory for Power Law Graphs

Jeremy Kepner, Nadya Bliss,
and Eric Robinson

MIT Lincoln Laboratory

This work is sponsored by the Department of Defense under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Government.

Outline

- **Introduction**
  - ***Goals***
  - ***Detection Theory***
  - ***Sparse Matrix Duality***
- **Backgrounds and foregrounds**
- **Tree Finding**
- **Summary**

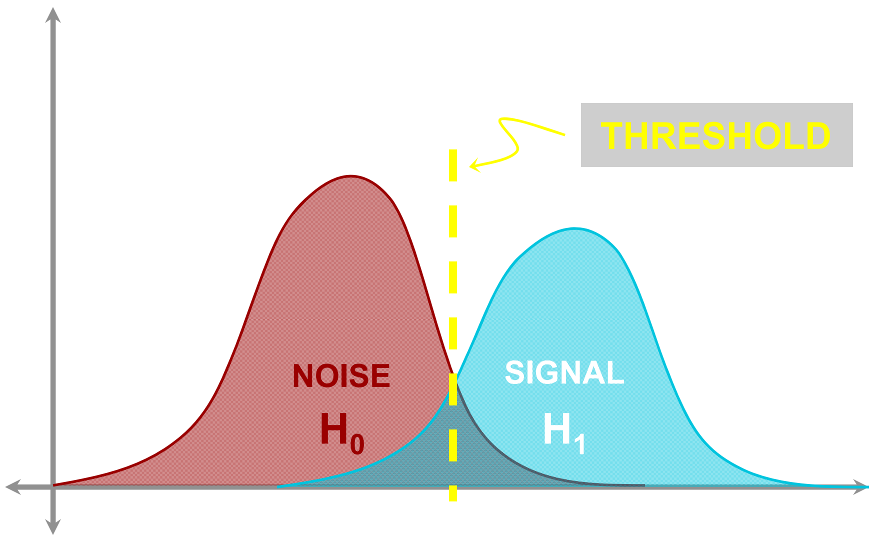
Goals

- **Detection Theory**
  - Apply basic postulates of detection theory (signal, background, …)
  - Quantitatively estimate difficulty of problem (SNR)
  - Develop better detection algorithms

- **Linear Algebraic Graph algorithms**
  - Additional tools for algorithm development
  - Compact representation
  - Parallel implementation well understood

Detection Theory

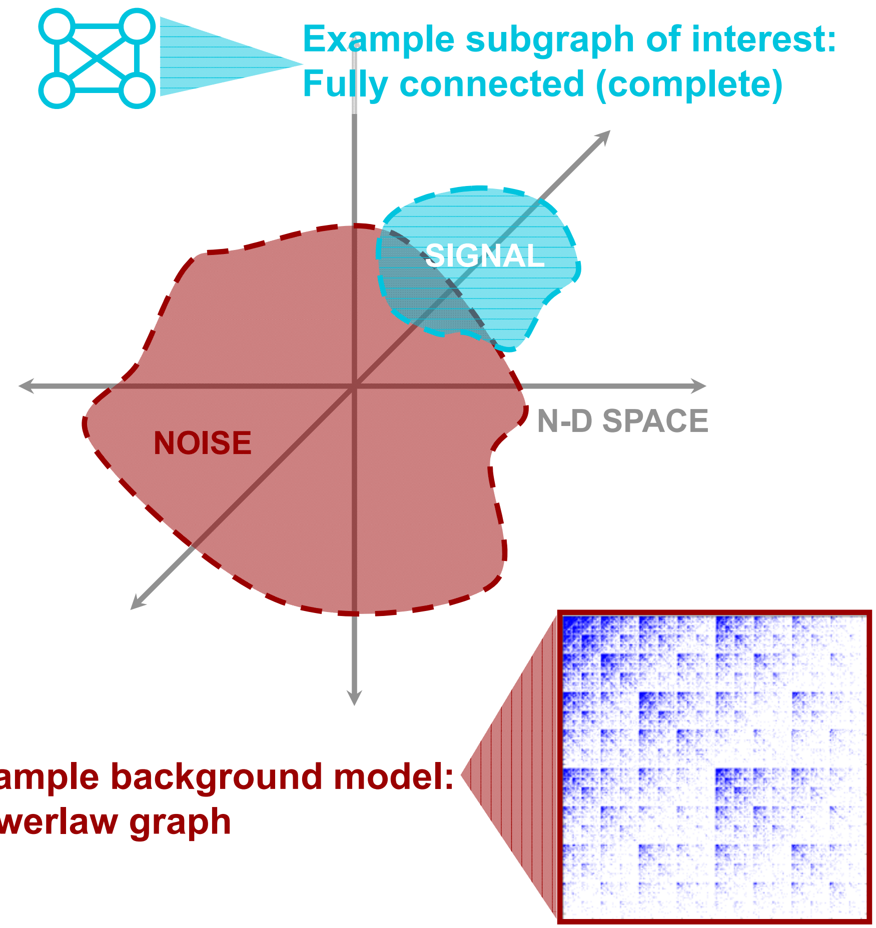
DETECTION OF SIGNAL IN NOISE



ASSUMPTIONS

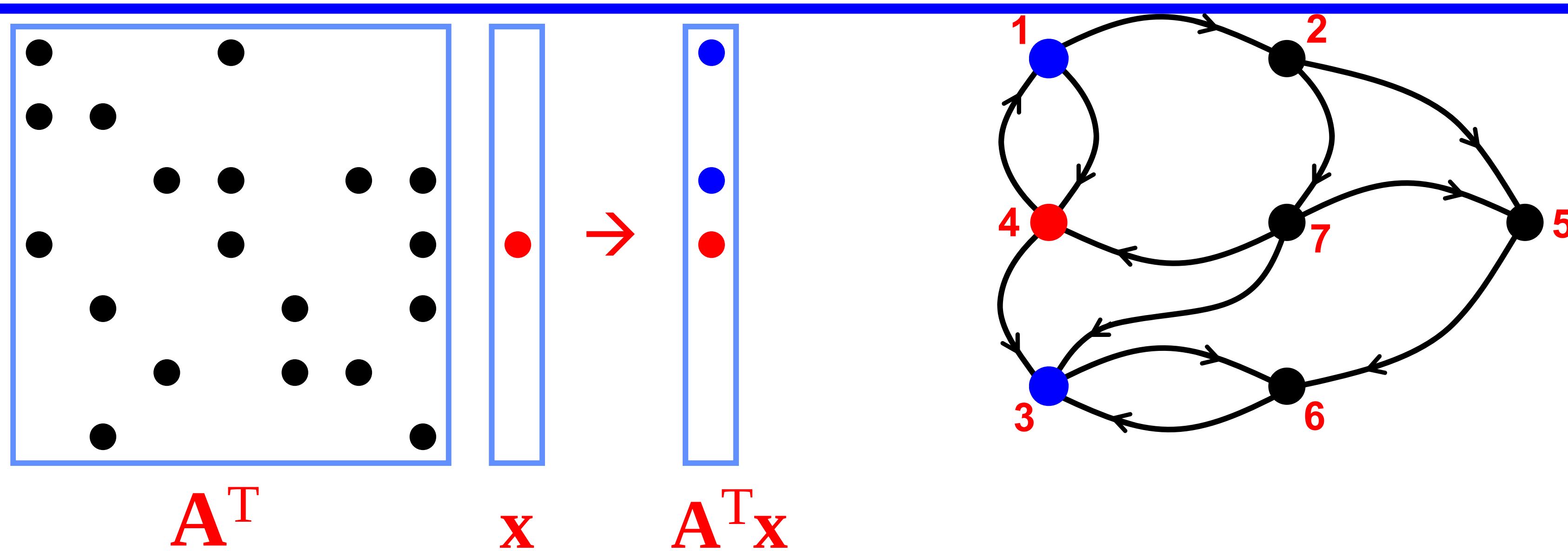
- Background (noise) statistics
- Foreground (signal) statistics
- Foreground/background separation
- Model ≈ reality

DETECTION OF SUBGRAPHS IN GRAPHS



Goal: Develop basic detection theory for finding subgraphs of interest in large background graphs

Graphs as Matrices



- Graphs can be represented as a sparse matrices
  - Multiply by adjacency matrix → step to neighbor vertices
  - Work-efficient implementation from sparse data structures
- Most algorithms reduce to products on semi-rings: C = A "+"."x" B
  - "x" : associative, distributes over "+"
  - "+" : associative, commutative
  - Examples: +.* min.+ or.and

Algorithm Comparison

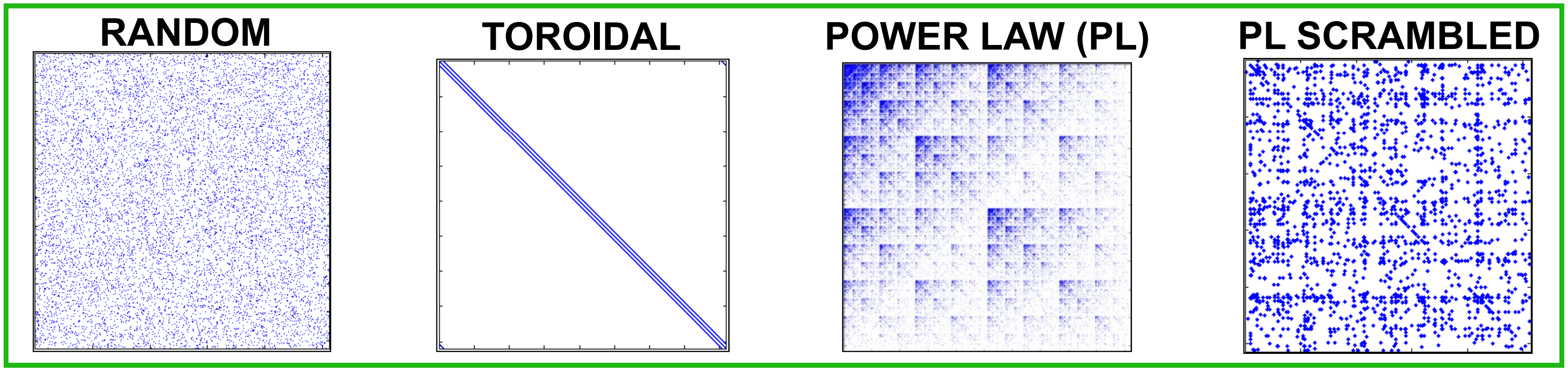
Algorithm (Problem)	Canonical Complexity	Array-Based Complexity	Critical Path (for array)
Bellman-Ford (SSSP)	$\Theta(mn)$	$\Theta(mn)$	$\Theta(n)$
Generalized B-F (APSP)	NA	$\Theta(n^3 \log n)$	$\Theta(\log n)$
Floyd-Warshall (APSP)	$\Theta(n^3)$	$\Theta(n^3)$	$\Theta(n)$
Prim (MST)	$\Theta(m+n \log n)$	$\Theta(n^2)$	$\Theta(n)$
Borůvka (MST)	$\Theta(m \log n)$	$\Theta(m \log n)$	$\Theta(\log^2 n)$
Edmonds-Karp (Max Flow)	$\Theta(m^2 n)$	$\Theta(m^2 n)$	$\Theta(mn)$
Push-Relabel (Max Flow)	$\Theta(mn^2)$ (or $\Theta(n^3)$)	$O(mn^2)$	?
Greedy MIS (MIS)	$\Theta(m+n \log n)$	$\Theta(mn+n^2)$	$\Theta(n)$
Luby (MIS)	$\Theta(m+n \log n)$	$\Theta(m \log n)$	$\Theta(\log n)$

Majority of selected algorithms can be represented with array-based constructs with equivalent complexity.

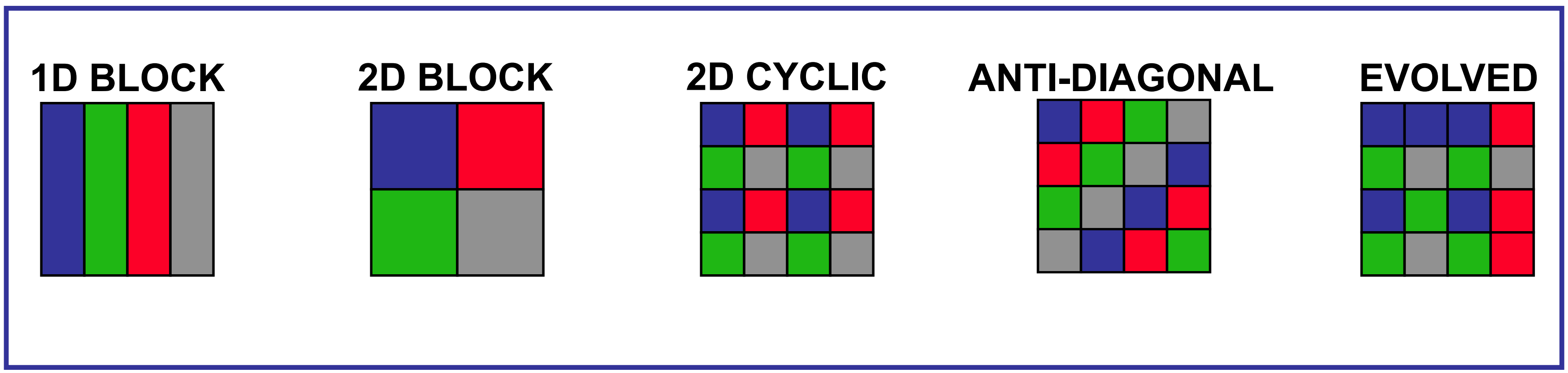
($n = |V|$ and $m = |E|$.)

Distributed Array Mapping

Adjacency Matrix Types:



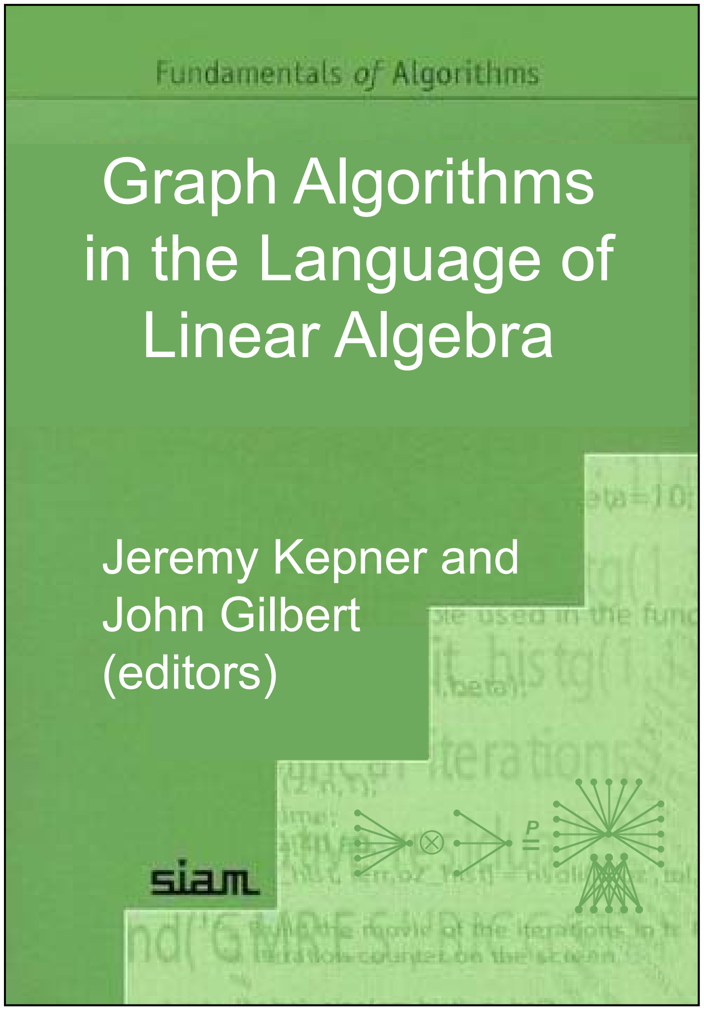
Distributions:



Sparse Matrix duality provides a natural way of exploiting distributed data distributions

Reference

- **Book: "Graph Algorithms in the Language of Linear Algebra"**
- **Editors: Kepner (MIT-LL) and Gilbert (UCSB)**
- **Contributors**
  - **Bader (Ga Tech)**
  - **Chakrabart (CMU)**
  - **Dunlavy (Sandia)**
  - **Faloutsos (CMU)**
  - **Fineman (MIT-LL & MIT)**
  - **Gilbert (UCSB)**
  - **Kahn (MIT-LL & Brown)**
  - **Kegelmeyer (Sandia)**
  - **Kepner (MIT-LL)**
  - **Kleinberg (Cornell)**
  - **Kolda (Sandia)**
  - **Leskovec (CMU)**
  - **Madduri (Ga Tech)**
  - **Robinson (MIT-LL & NEU), Shah (UCSB)**

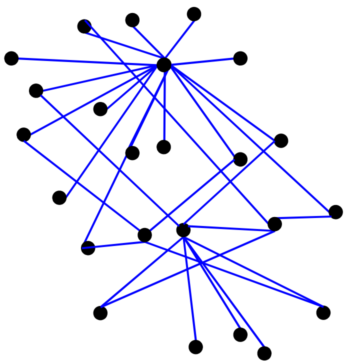


Outline

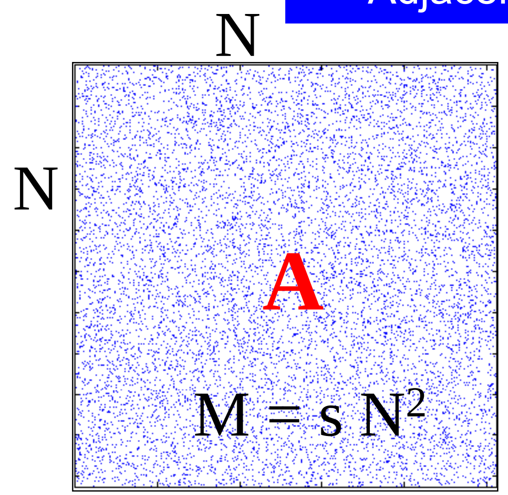
- Introduction
- **Background and foregrounds**
  - *Random*
  - *Power Law*
  - *Clique*
  - *Source/Sink*
  - *Tree*
- Tree Finding
- Summary

Background: Random (Erdos-Renyi)

Graph



Adjacency Matrix



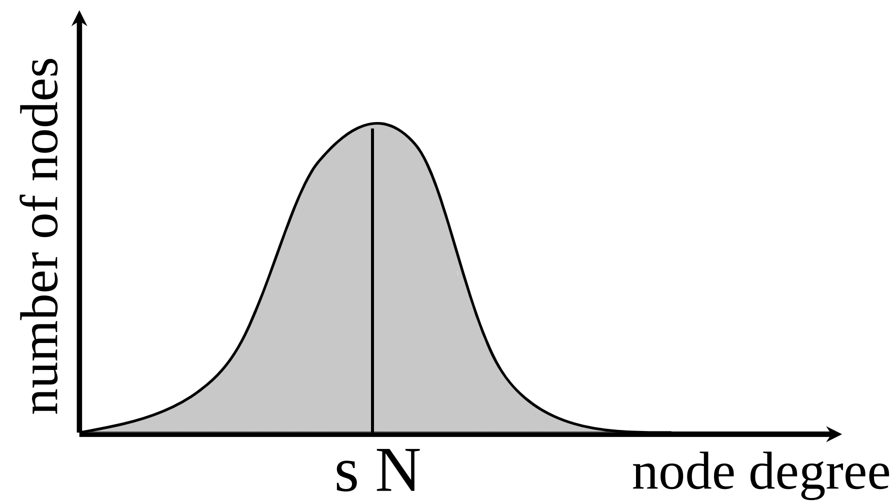
Algebraic Form

$$\mathbf{A} : \mathbb{B}^{N \times N}$$

$$\mathbf{A}(i,j) = (r < s)$$

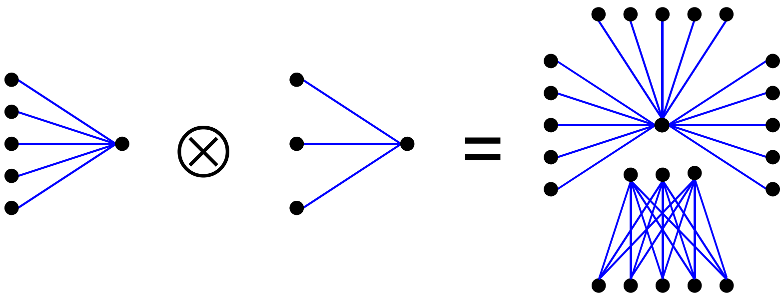
$$r \leftarrow [0,1]$$

Degree Distribution

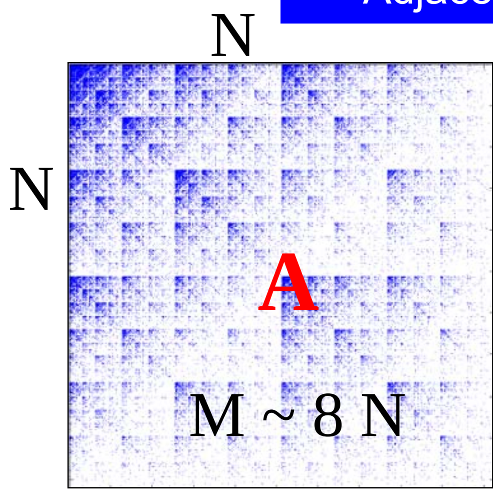


Background: Power Law (Kronecker)

Graph



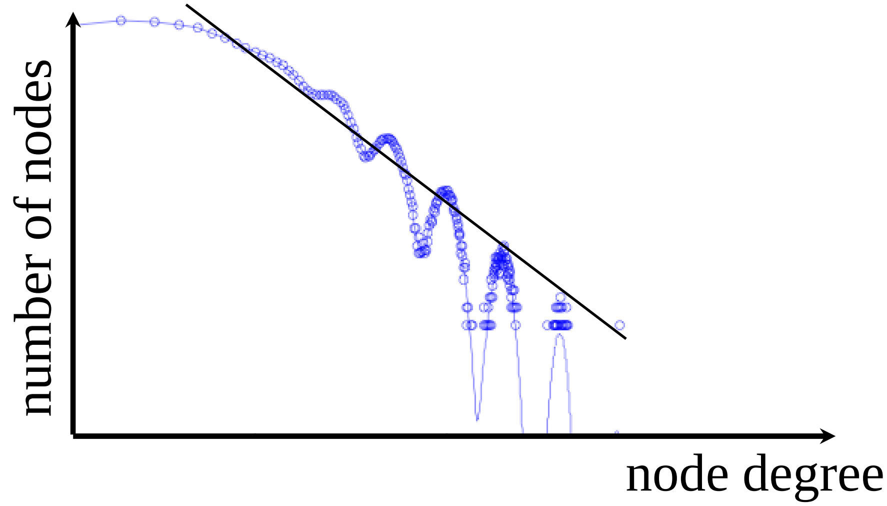
Adjacency Matrix



$$\mathbf{G} : \mathbf{R}^{n \times n}$$

$$\mathbf{A} \overset{M}{\leftarrow} \mathbf{G}^{\otimes k} = \mathbf{G}^{\otimes k-1} \otimes \mathbf{G}$$

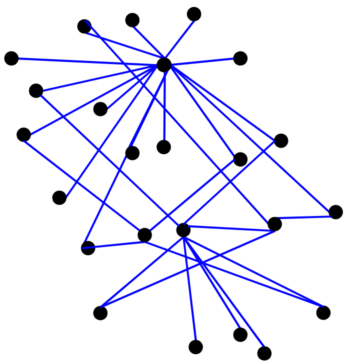
Algebraic Form



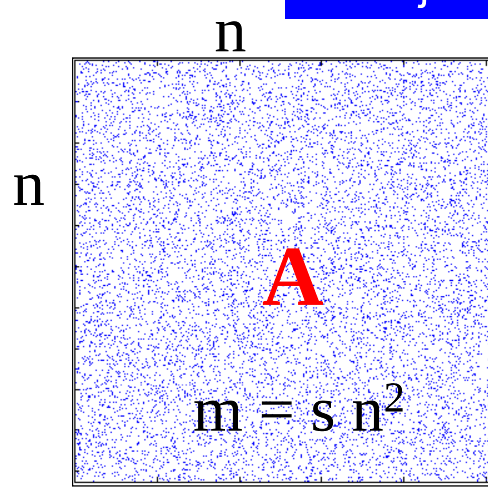
Degree Distribution

Foreground: Clique (Partial)

Graph



Adjacency Matrix



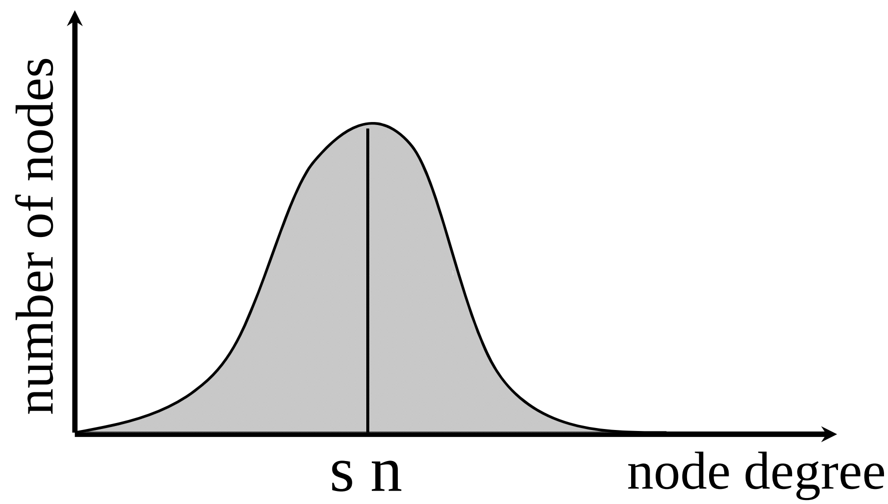
Algebraic Form

$$\mathbf{A} : \mathbb{B}^{n \times n}$$

$$\mathbf{A}(i,j) = (r < s)$$

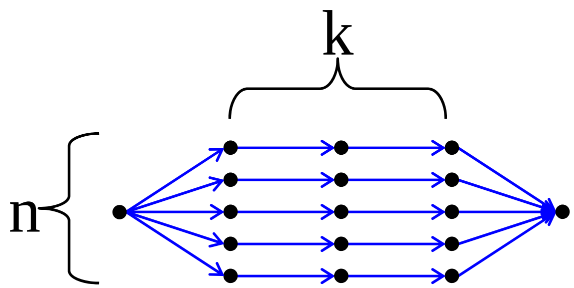
$$r \leftarrow [0,1]$$

Degree Distribution

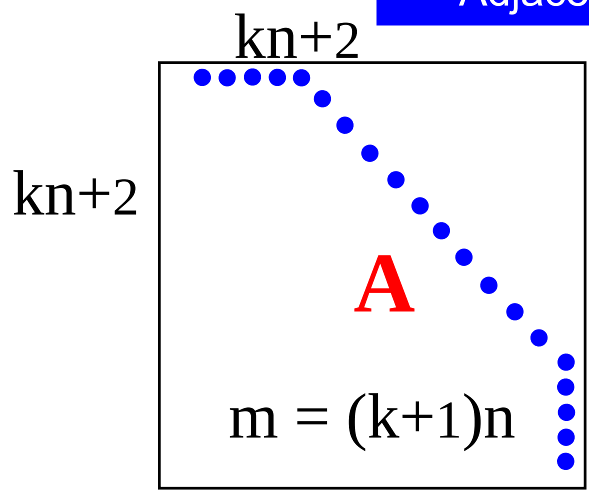


Foreground: Source Sink

Graph



Adjacency Matrix

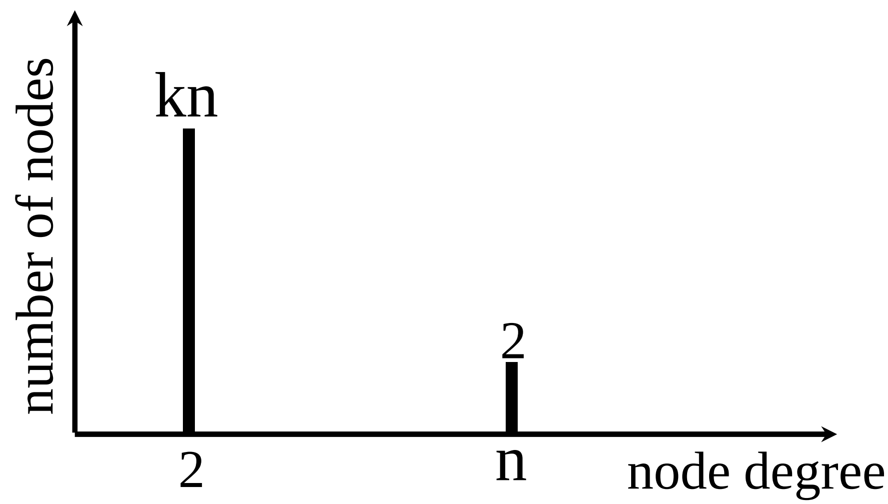


Algebraic Form

$$\mathbf{A} = \overset{k\times k}{\begin{pmatrix}0 & 1 & & \\ & \ddots & \ddots & \\ & & & 1 \\ & & & 0\end{pmatrix}} \otimes \overset{n\times n}{\begin{pmatrix}1 & & \\ & \ddots & \\ & & 1\end{pmatrix}} +$$

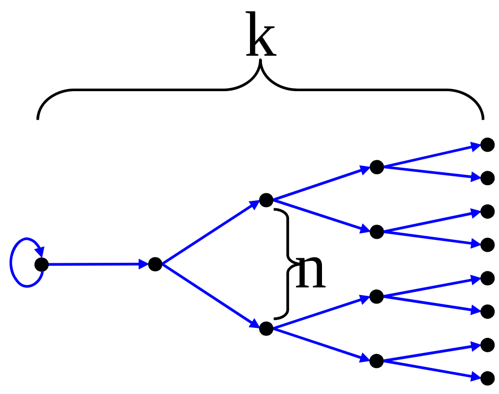
$$\underset{k\times k}{\begin{pmatrix}1 & & \\ & & \\ & & \end{pmatrix}} \otimes \underset{1\times n}{\begin{pmatrix}1\end{pmatrix}} \otimes \underset{n\times 1}{\begin{pmatrix}1 \\ \vdots \\ 1\end{pmatrix}'} + \underset{k\times k}{\begin{pmatrix} & & \\ & & \\ & & 1\end{pmatrix}} \otimes \underset{n\times 1}{\begin{pmatrix}1\end{pmatrix}'} \otimes \underset{1\times n}{\begin{pmatrix}1 \\ \vdots \\ 1\end{pmatrix}}$$

Degree Distribution

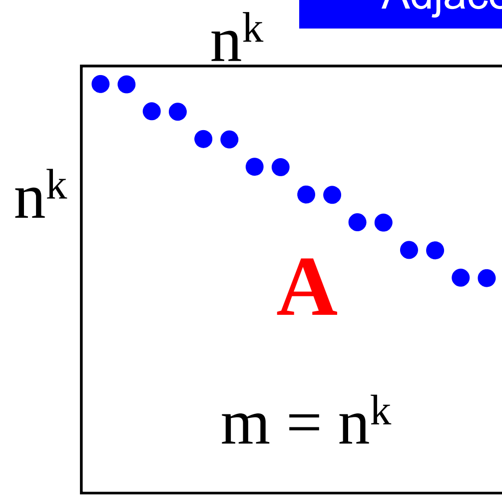


Foreground: Trees

Graph



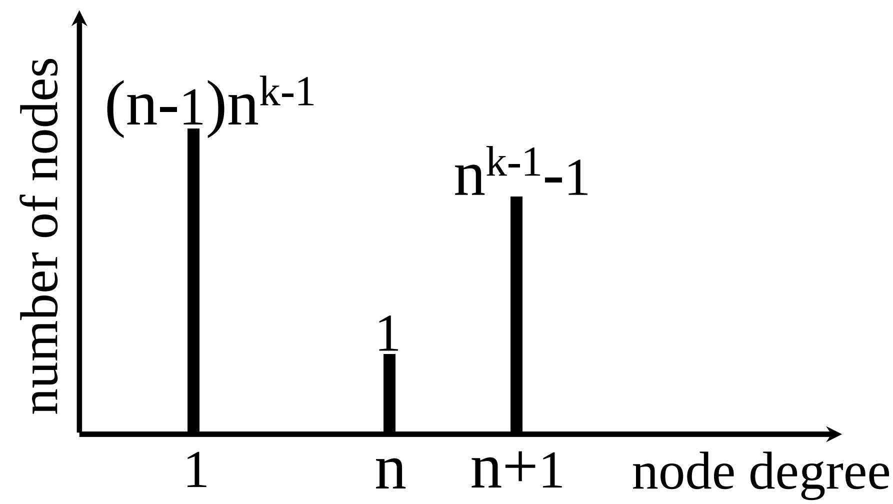
Adjacency Matrix



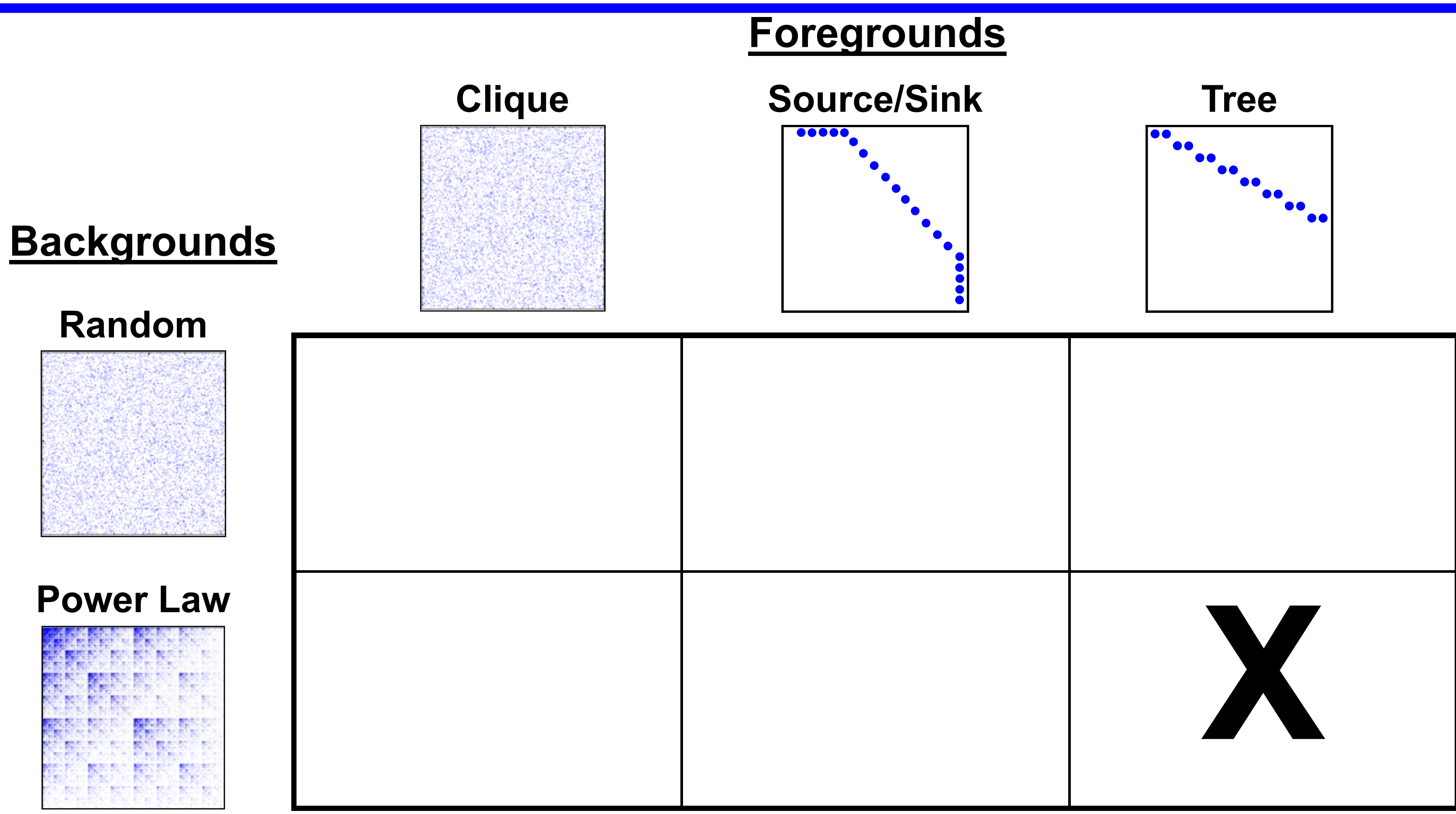
Algebraic Form

$$\mathbf{A} = \underset{n\times 1}{\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}} \otimes \underset{n\times n}{\begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix}}^{\otimes k-1} \otimes \underset{1\times n}{\begin{pmatrix} 1 & \cdots & 1 \end{pmatrix}}$$

Degree Distribution



Background/Foreground Combinations

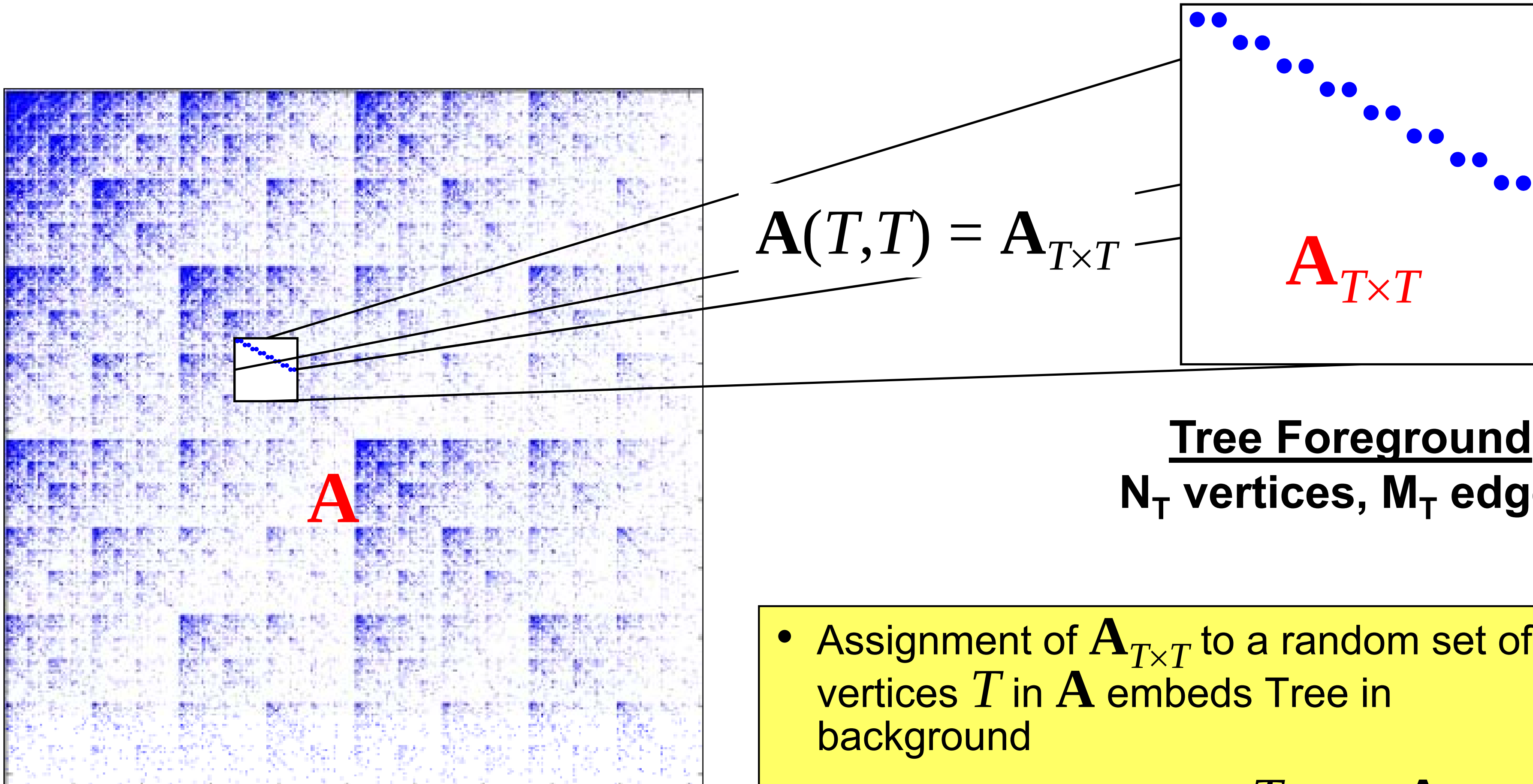


- Many interesting background/foreground combinations
- Rest of talk will focus on power law/tree

Outline

- Introduction
- Background and foregrounds
- **Tree Finding**
  - *Embedding*
  - *Cued vs Uncued*
  - *Set-Vector Representation*
  - *Algorithm*
  - *Results*
- Summary

Tree Embedding



Power Law Background
N vertices, M edges

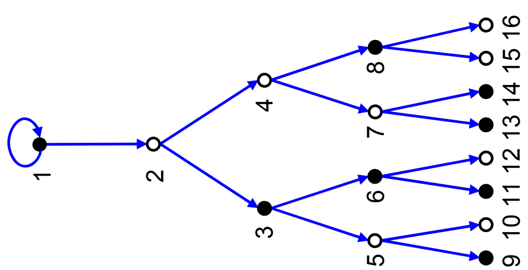
Tree Foreground
N_T vertices, M_T edges

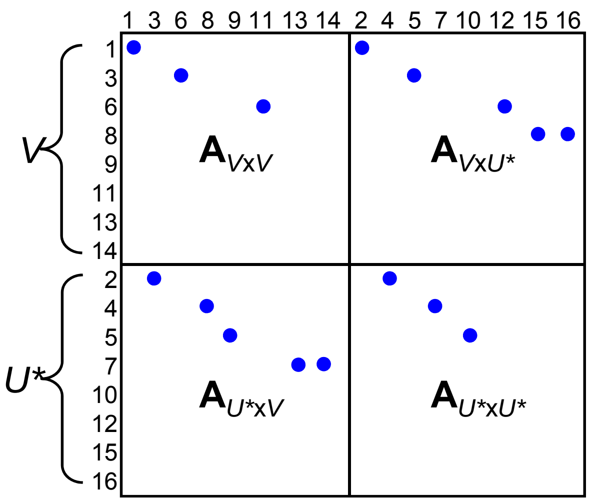
- Assignment of $\mathbf{A}_{T\times T}$ to a random set of vertices T in $\mathbf{A}$ embeds Tree in background
- Detection problem: find T given $\mathbf{A}$
  - Assume $N >> N_T$ and $M >> M_T$

Cued vs. Uncued Detection

- Uncued detection
  - No information about T is provided
  - Signal-to-noise ratio ~ N_T/N
  - Extremely difficult

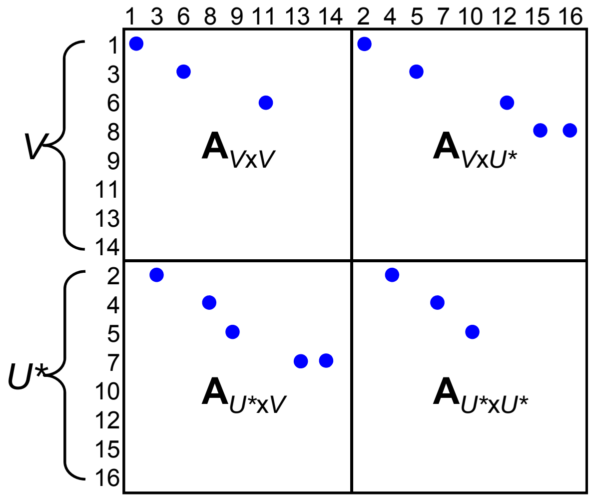
- Cued detection
  - T is divided into two sets V (given) and U^* (unknown)
  - More tractable



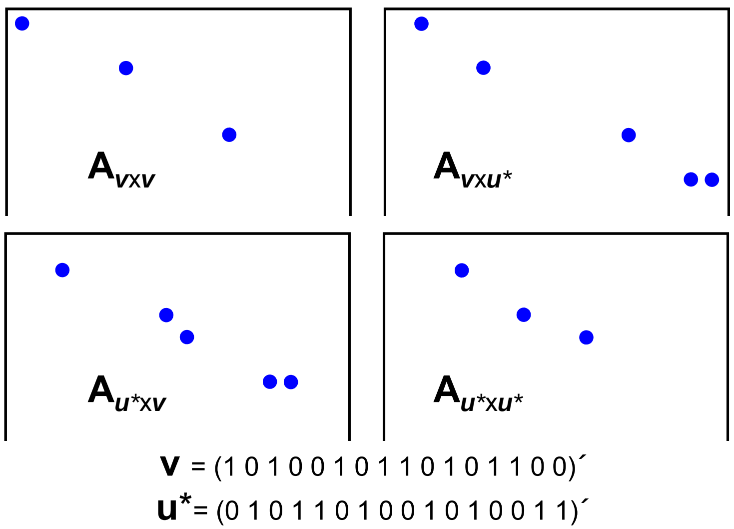


Set-Vector Dual Representation

Set Representation



Vector Representation



- Set of vertices V can also be represented as an N element vector where $\mathbf{v}(V) = 1$, allows multiple adjacency matrix representations

$$\mathbf{A}_{V \times V} = \mathbf{A}(V,V) \quad \text{or} \quad \mathbf{A}_{\mathbf{v}\times\mathbf{v}} = \mathbf{I}_{\mathbf{v}}\,\mathbf{A}\,\mathbf{I}_{\mathbf{v}}$$

- Set representation better for visualization
  – V contains only elements of interest
- Vector better for algorithm development and implementation
  – $\mathbf{v}$ allows linear algebraic transformations and preserves graph context

Tree Finding Algorithm Summary

- Step 0: Find all vertices that are 1st neighbors of $\mathbf{v}$

$$\mathbf{A}_{\mathbf{u}_0 \times \mathbf{v}} = \mathbf{A}\,\mathbf{I}_{\mathbf{v}} - \mathbf{A}_{\mathbf{v} \times \mathbf{v}}$$

$$\mathbf{d}_{\mathbf{u}_0 \times \mathbf{v}} = \mathbf{A}_{\mathbf{u}_0 \times \mathbf{v}} + \mathbf{A}'_{\mathbf{v} \times \mathbf{u}_0}$$

$$\mathbf{u}_0 = \mathbf{d}_{\mathbf{u}_0 \times \mathbf{v}} > 0$$

- Step 1a: Eliminate vertices that create too many connections to $\mathbf{v}$
- Step 1b: Eliminate vertices that connect to $\mathbf{v}$ that are filled
- Step 2: Find all vertices that are 1st neighbors of $\mathbf{v}$ that satisfy 1a & 1b
- Step 3: Select highest probability vertices based on (edges available) / (number candidates)
- Step 4: Select vertices with multiple connections into $\mathbf{v}$

Signal-to-Noise Estimate

- **Background power law: N = 2^{20}**
- **Foreground binary tree $N_T = 2^7$, f = 0.5 (fraction known)**

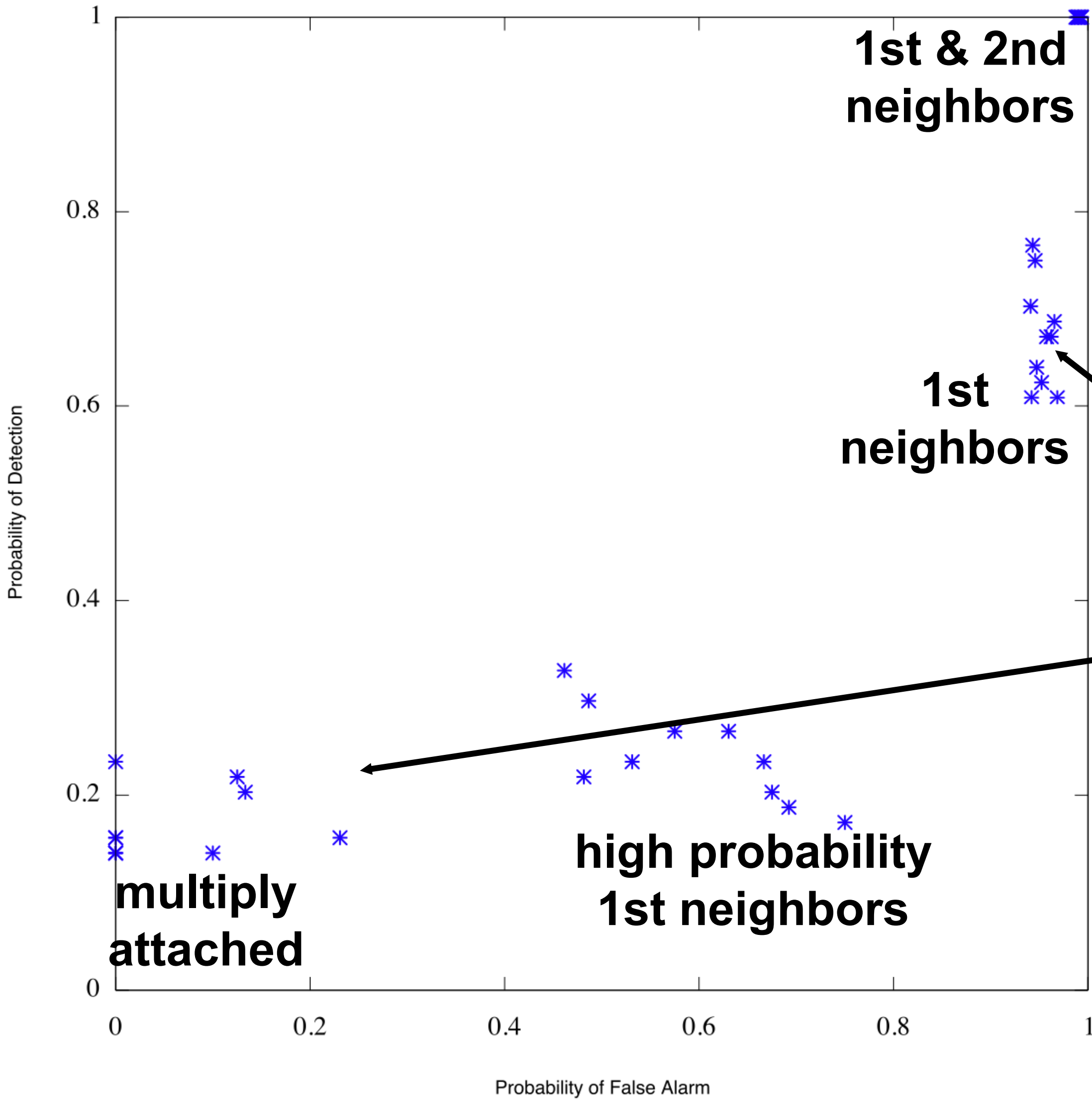
- **Baseline SNR ~ 2^{-14} ~ 0.00006**

- **1st and 2nd neighbors SNR ~ $5/2^{12}$ ~ 0.001**

- **1st neighbors SNR ~ $7/2^8$ ~ 0.03**
  - **Step 0**

- **Multiply attached neighbors SNR ~ 2^4 ~ 16**
  - **Step 4**

Probability of Detection (PD) vs Probability of False Alarm (PFA)



- PD vs PFA consistent with SNR estimates
- Can find most of tree with high PFA
- Can find subset of tree with very low PFA

Summary

- **Detection Theory**
  - Apply basic postulates of detection theory (signal, background, …)
  - Quantitatively estimate difficulty of problem (SNR)
  - Develop better detection algorithms

- **Linear Algebraic Graph algorithms**
  - Additional tools for algorithm development
  - Compact representation
  - Parallel implementation well understood