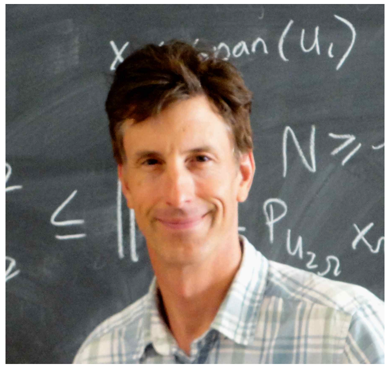
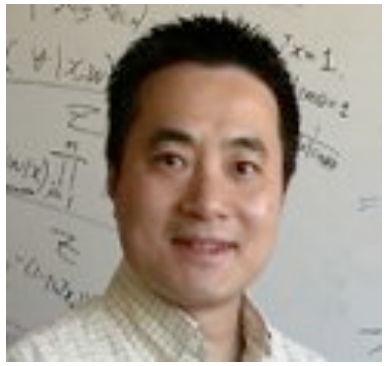
S^2: Efficient Graph Based Active Learning

Gautam Dasarathy
Machine Learning,
Carnegie Mellon University
gautamd@cs.cmu.edu
http://gautamdasarathy.com

joint work with



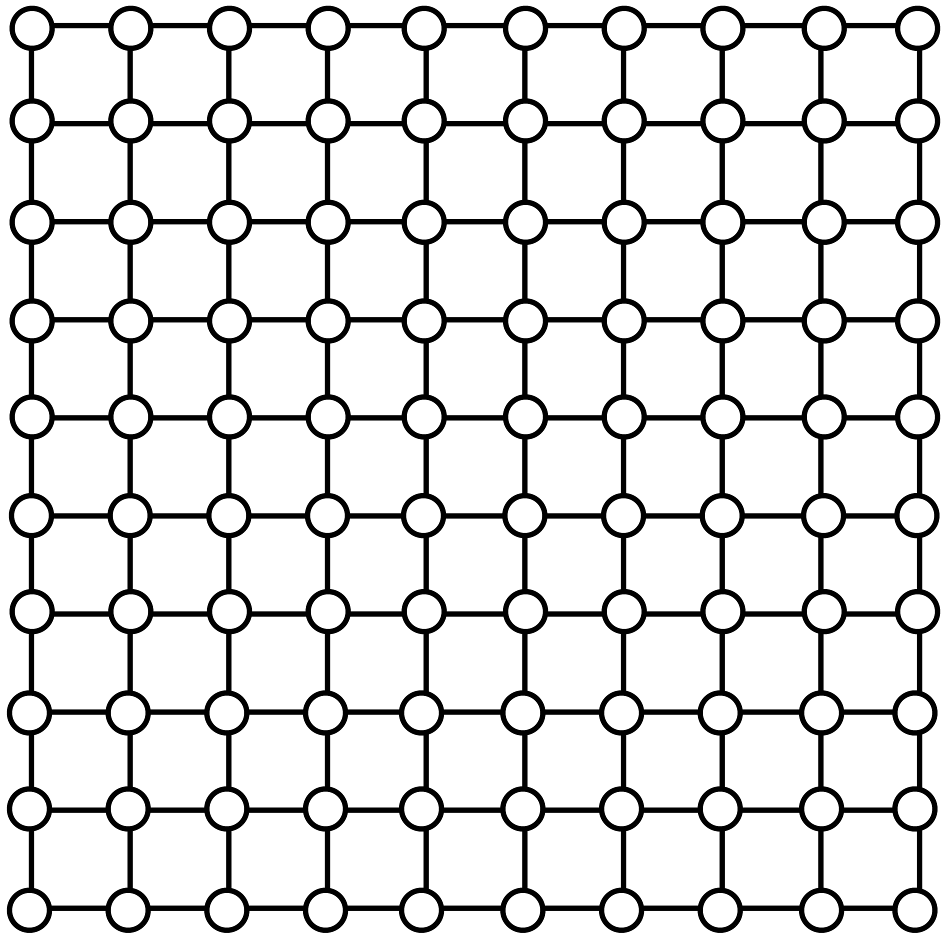
Rob Nowak
ECE
UW - Madison



Xiaojin (Jerry) Zhu
Computer Sciences
UW - Madison

The Problem

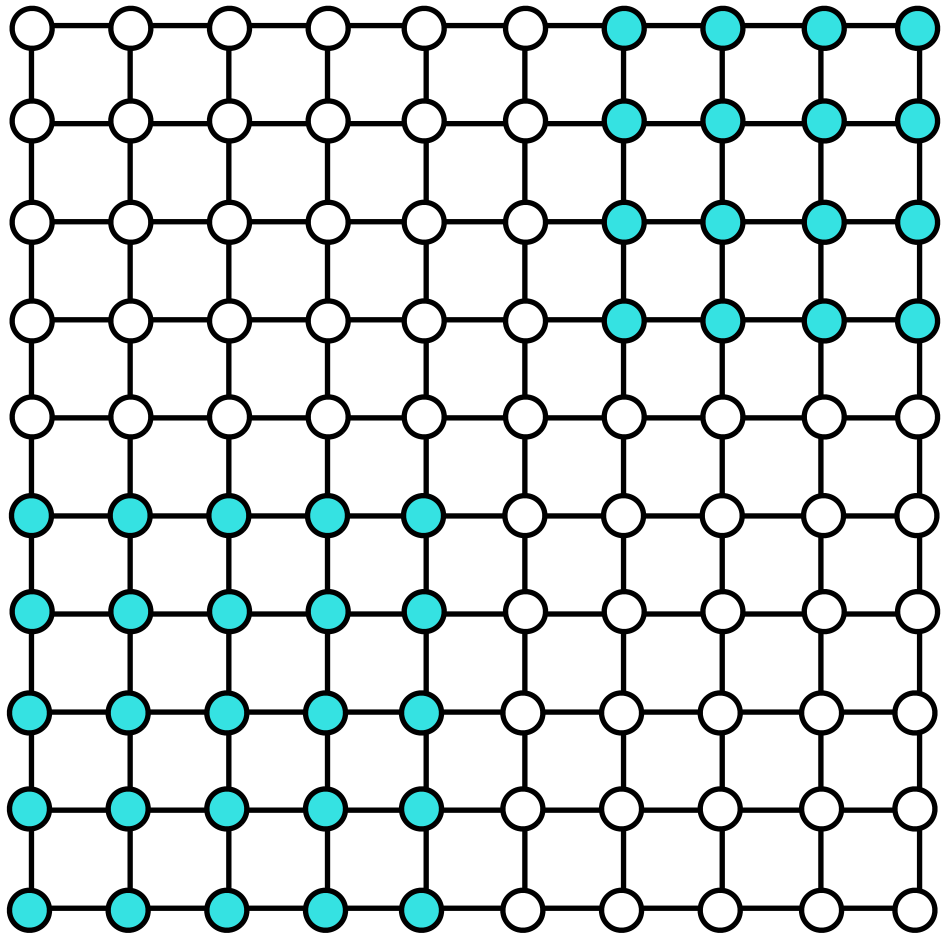
$G = ([n], E)$ is a known **undirected graph** on $[n] = \{1, 2, \ldots, n\}$



The Problem

$G = ([n], E)$ is a known **undirected graph** on $[n] = \{1, 2, \ldots, n\}$

$f : [n] \to \{-1, +1\}$ is an **unknown labeling function** that we want to learn.



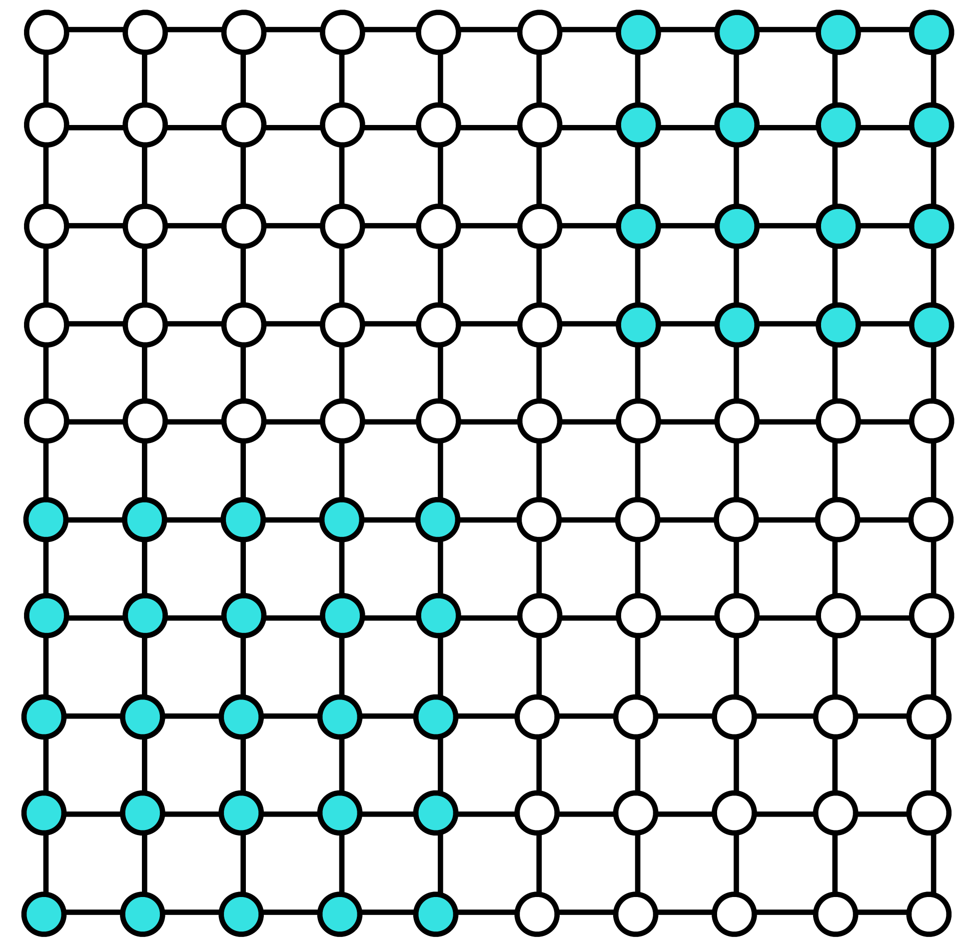
The Problem

$G = ([n], E)$ is a known **undirected graph** on $[n] = \{1, 2, \ldots, n\}$

$f : [n] \to \{-1, +1\}$ is an **unknown labeling function** that we want to learn.

Goal: **Sequentially** and **actively** select a subset $L \subset [n]$ to be labeled.

Predict $\{f(i) : i \notin L\}$.



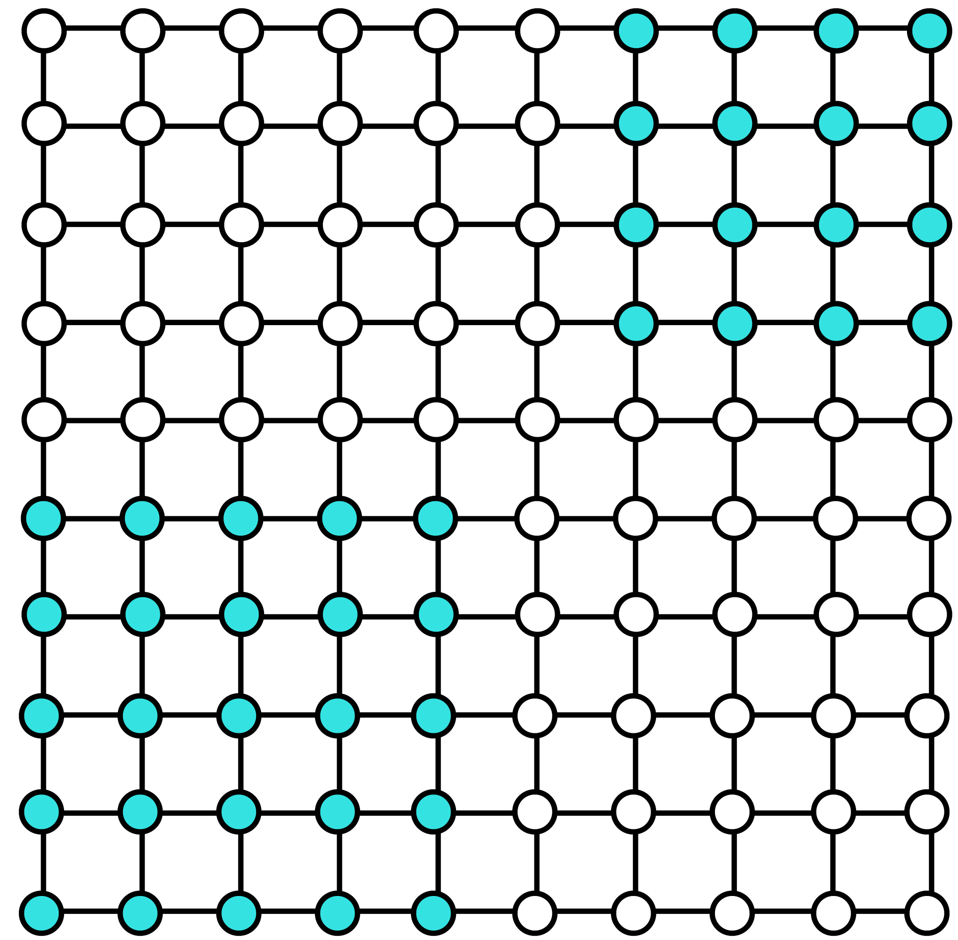
The Problem

$G = ([n], E)$ is a known **undirected graph** on $[n] = \{1, 2, \ldots, n\}$

$f : [n] \to \{-1, +1\}$ is an **unknown labeling function** that we want to learn.

Goal: **Sequentially** and **actively** select a subset $L \subset [n]$ to be labeled.

Predict $\{f(i) : i \notin L\}$.

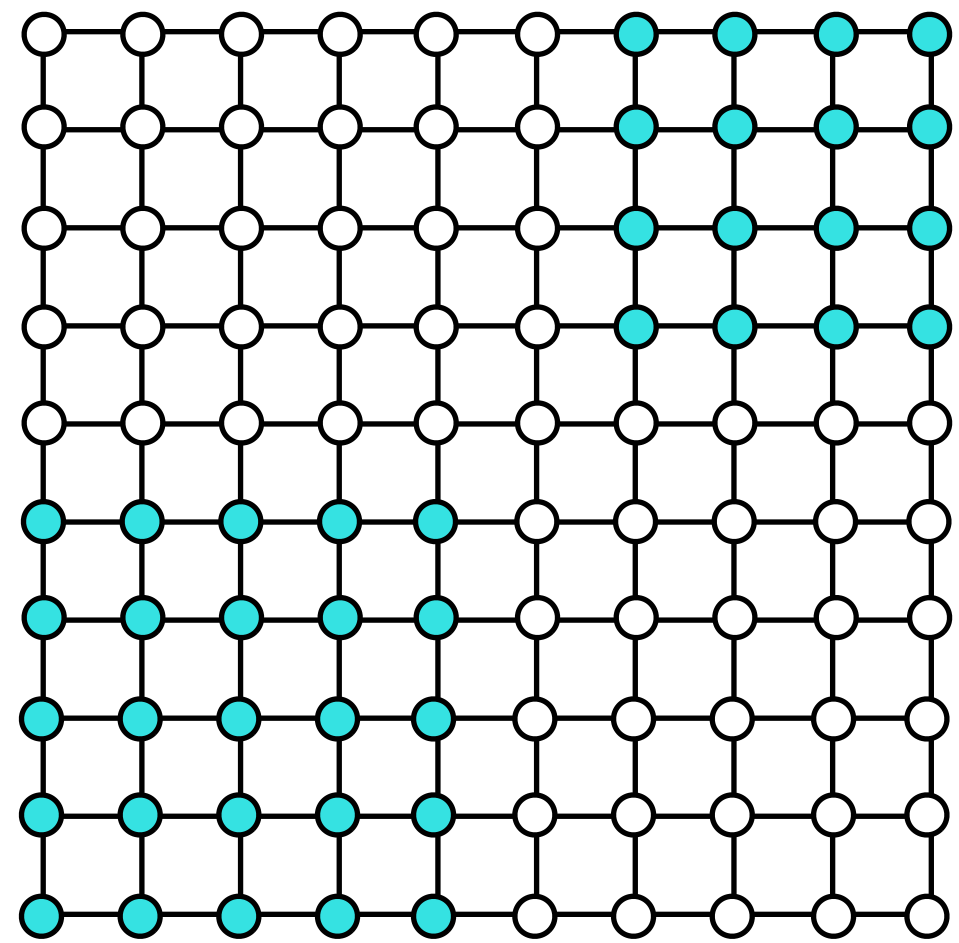


We are naturally interested in questions like:

- How to choose *L* (efficiently) ?
- How big does L have to be ?
- How does the interaction between f and G affect these things?

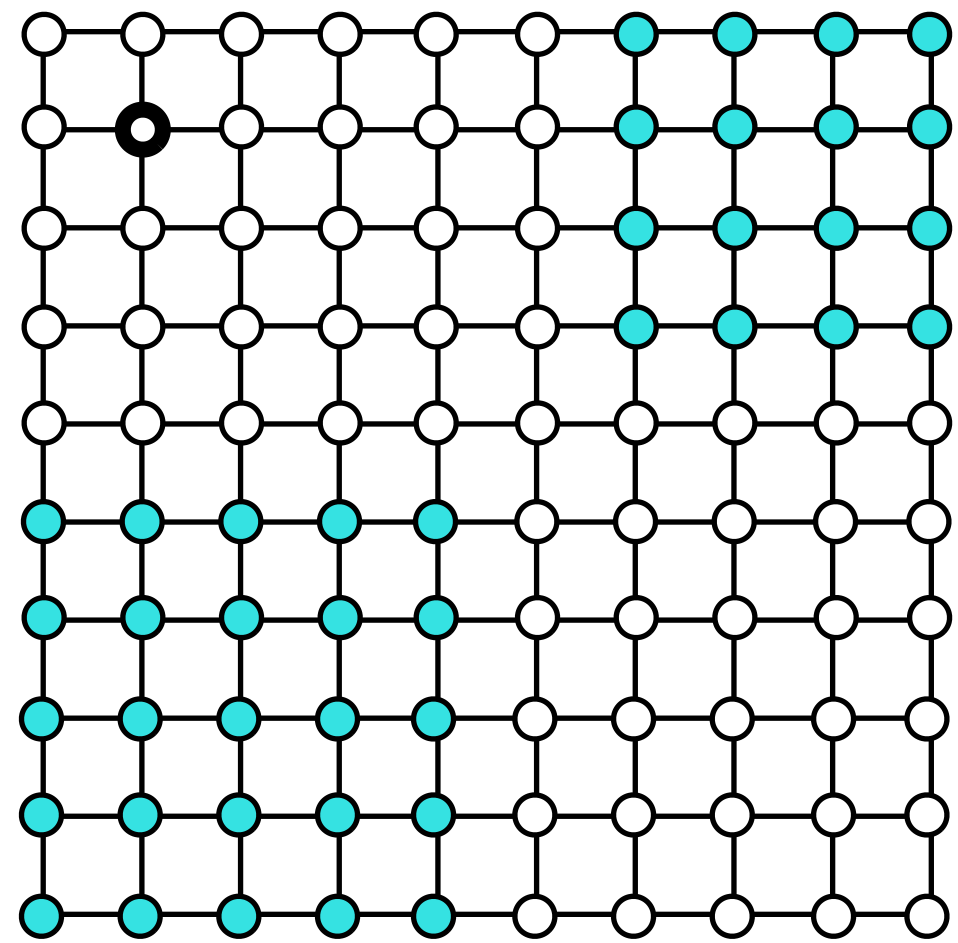
The S^2 Algorithm

- Randomly query vertices till you find a pair with opposite labels.
- Among all shortest paths connecting oppositely labeled vertices, pick the shortest one and query the vertex at its mid-point.
  - If you find a "cut" edge, remove it and proceed.
- If there are no more oppositely labeled vertices, go back to step 1 (random sampling).



The S^2 Algorithm

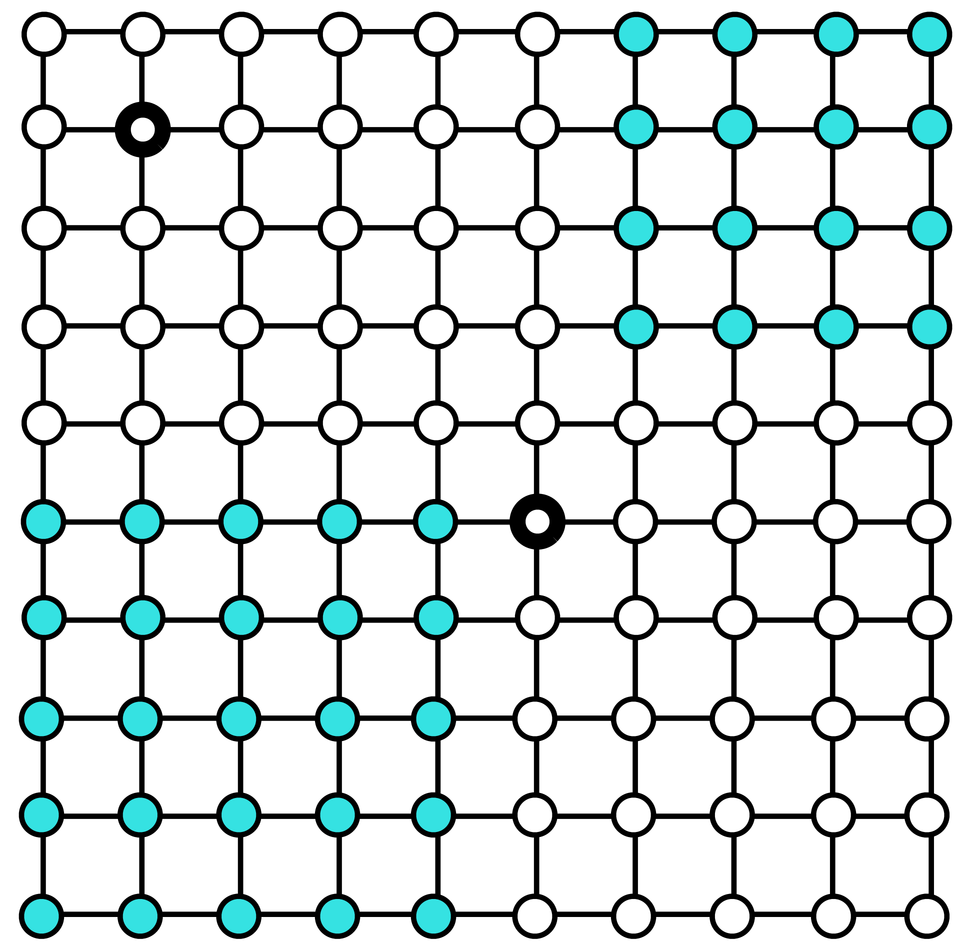
- Randomly query vertices till you find a pair with opposite labels.
- Among all shortest paths connecting oppositely labeled vertices, pick the shortest one and query the vertex at its mid-point.
  - If you find a "cut" edge, remove it and proceed.
- If there are no more oppositely labeled vertices, go back to step 1 (random sampling).



Randomly query

The S^2 Algorithm

- Randomly query vertices till you find a pair with opposite labels.
- Among all shortest paths connecting oppositely labeled vertices, pick the shortest one and query the vertex at its mid-point.
  - If you find a "cut" edge, remove it and proceed.
- If there are no more oppositely labeled vertices, go back to step 1 (random sampling).



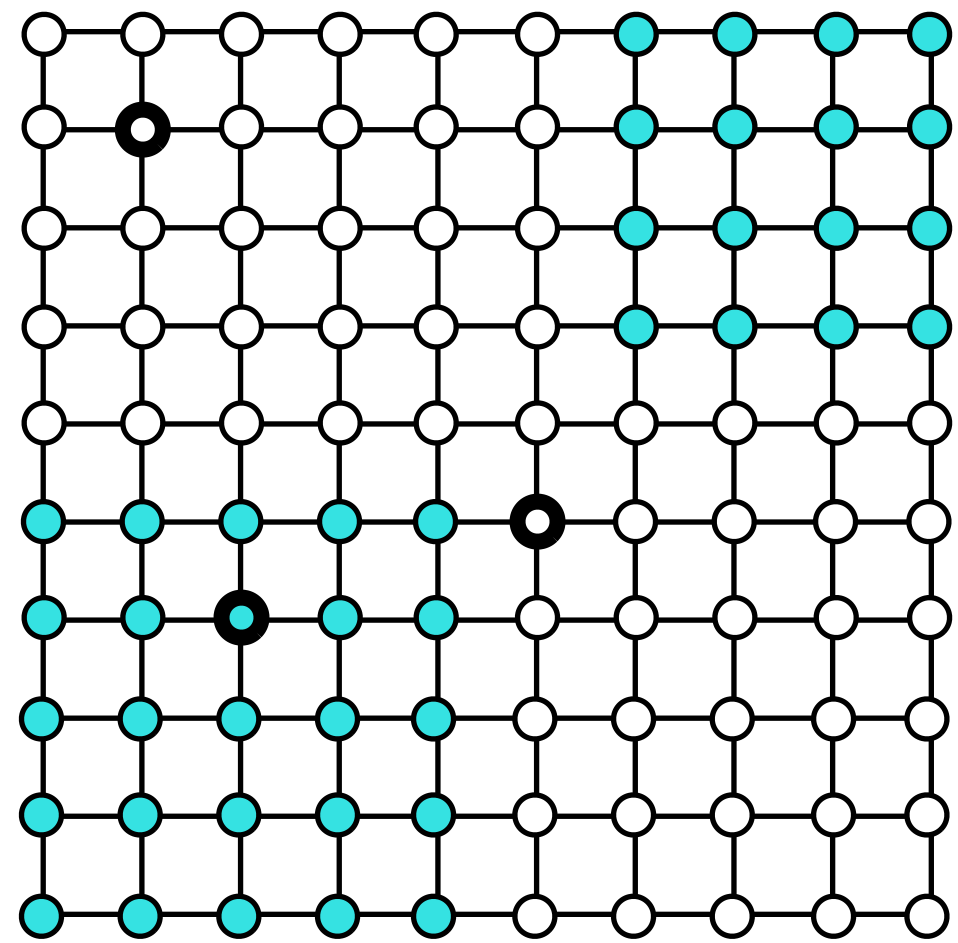
Randomly query

The S^2 Algorithm

- Randomly query vertices till you find a pair with opposite labels.

- Among all shortest paths connecting oppositely labeled vertices, pick the shortest one and query the vertex at its mid-point.
  - If you find a "cut" edge, remove it and proceed.

- If there are no more oppositely labeled vertices, go back to step 1 (random sampling).



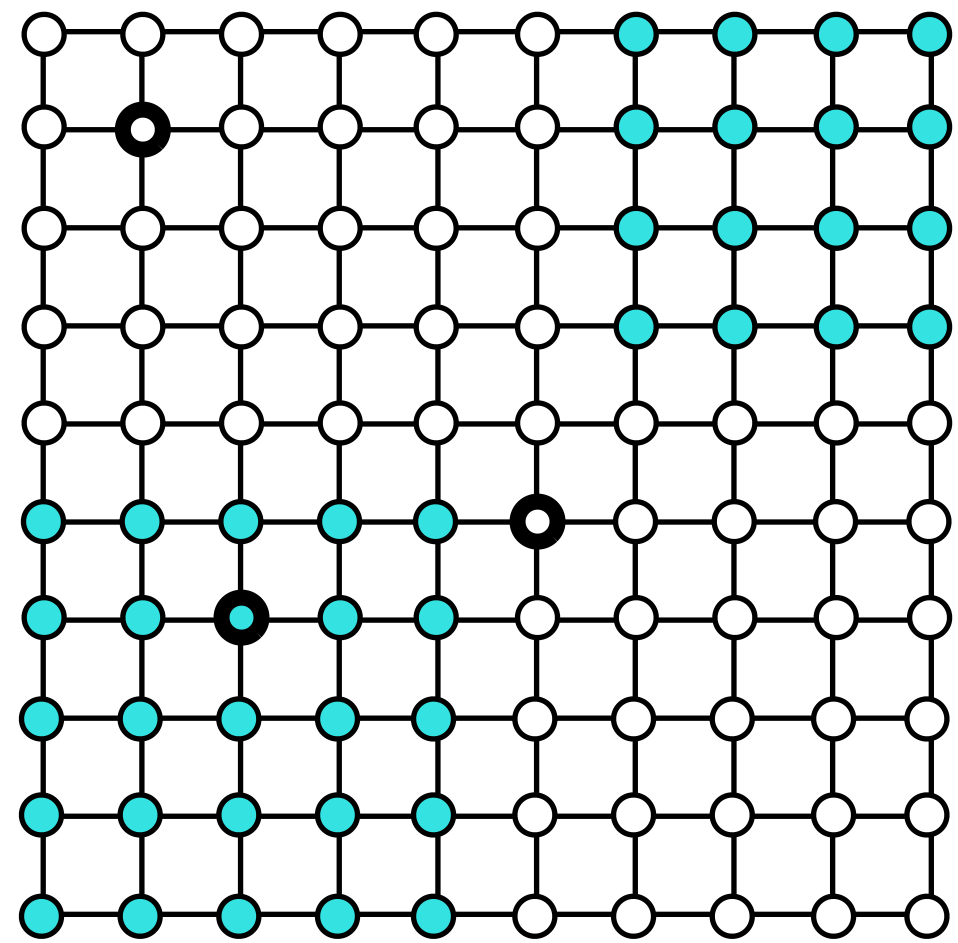
Randomly query

The S^2 Algorithm

- Randomly query vertices till you find a pair with opposite labels.

- Among all shortest paths connecting oppositely labeled vertices, pick the shortest one and query the vertex at its mid-point.
  - If you find a "cut" edge, remove it and proceed.

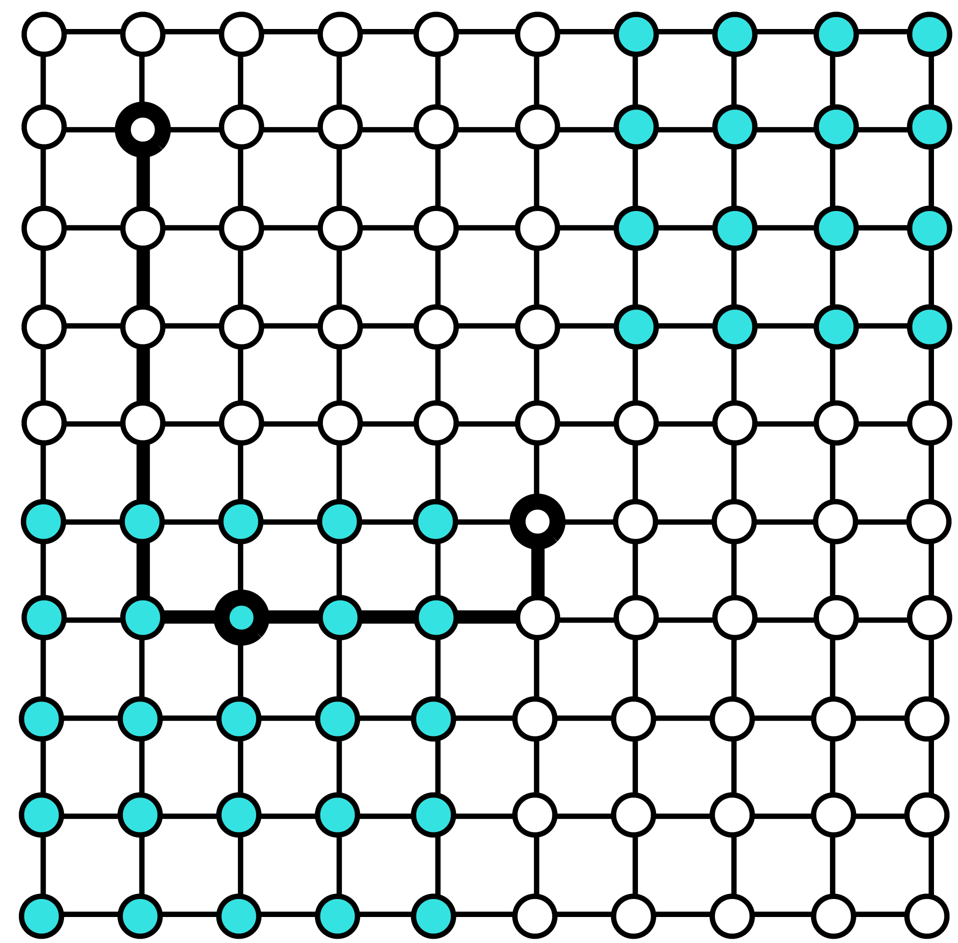
- If there are no more oppositely labeled vertices, go back to step 1 (random sampling).



Found oppositely labeled vertices

The S^2 Algorithm

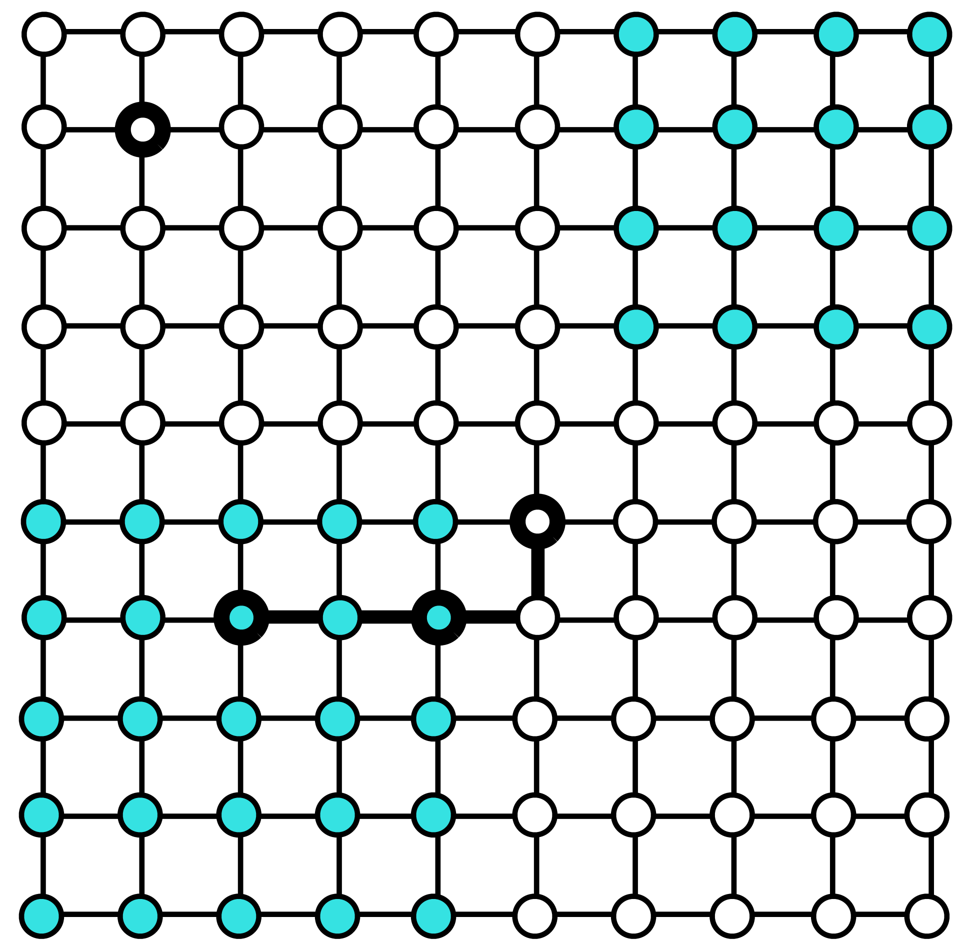
- Randomly query vertices till you find a pair with opposite labels.
- Among all shortest paths connecting oppositely labeled vertices, pick the shortest one and query the vertex at its mid-point.
  - If you find a "cut" edge, remove it and proceed.
- If there are no more oppositely labeled vertices, go back to step 1 (random sampling).



There are 2 shortest paths connecting oppositely labeled vertices.

The S^2 Algorithm

- Randomly query vertices till you find a pair with opposite labels.
- Among all shortest paths connecting oppositely labeled vertices, pick the shortest one and query the vertex at its mid-point.
  - If you find a "cut" edge, remove it and proceed.
- If there are no more oppositely labeled vertices, go back to step 1 (random sampling).



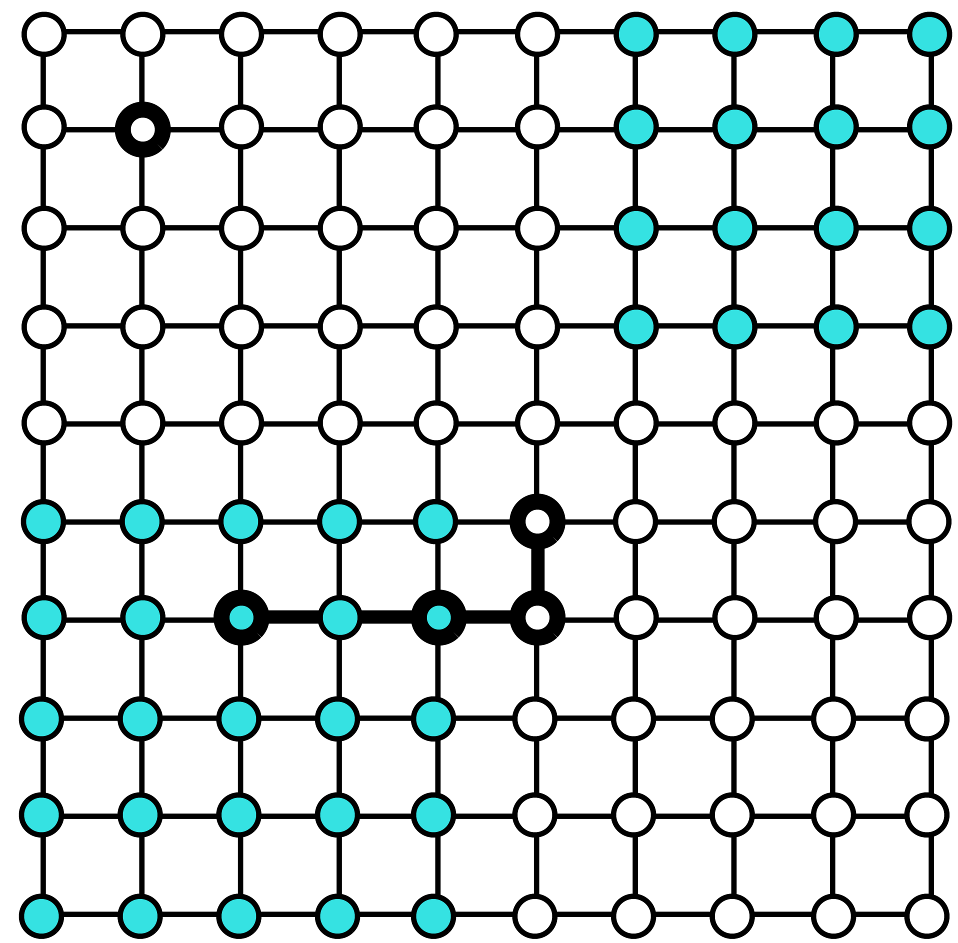
Pick the shortest shortest path and query at midpoint.

The S^2 Algorithm

- Randomly query vertices till you find a pair with opposite labels.

- Among all shortest paths connecting oppositely labeled vertices, pick the shortest one and query the vertex at its mid-point.
  - If you find a "cut" edge, remove it and proceed.

- If there are no more oppositely labeled vertices, go back to step 1 (random sampling).



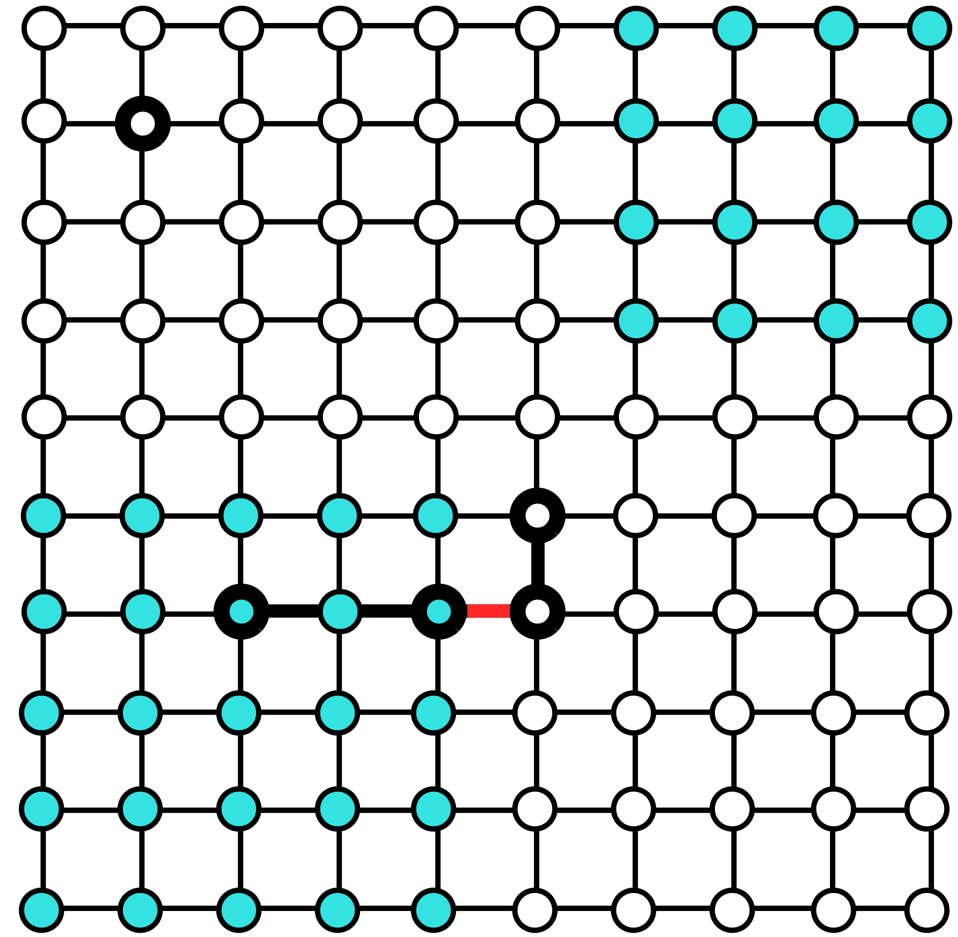
Continue this and remove any cut edges found

The S^2 Algorithm

- Randomly query vertices till you find a pair with opposite labels.

- Among all shortest paths connecting oppositely labeled vertices, pick the shortest one and query the vertex at its mid-point.
  - If you find a "cut" edge, remove it and proceed.

- If there are no more oppositely labeled vertices, go back to step 1 (random sampling).



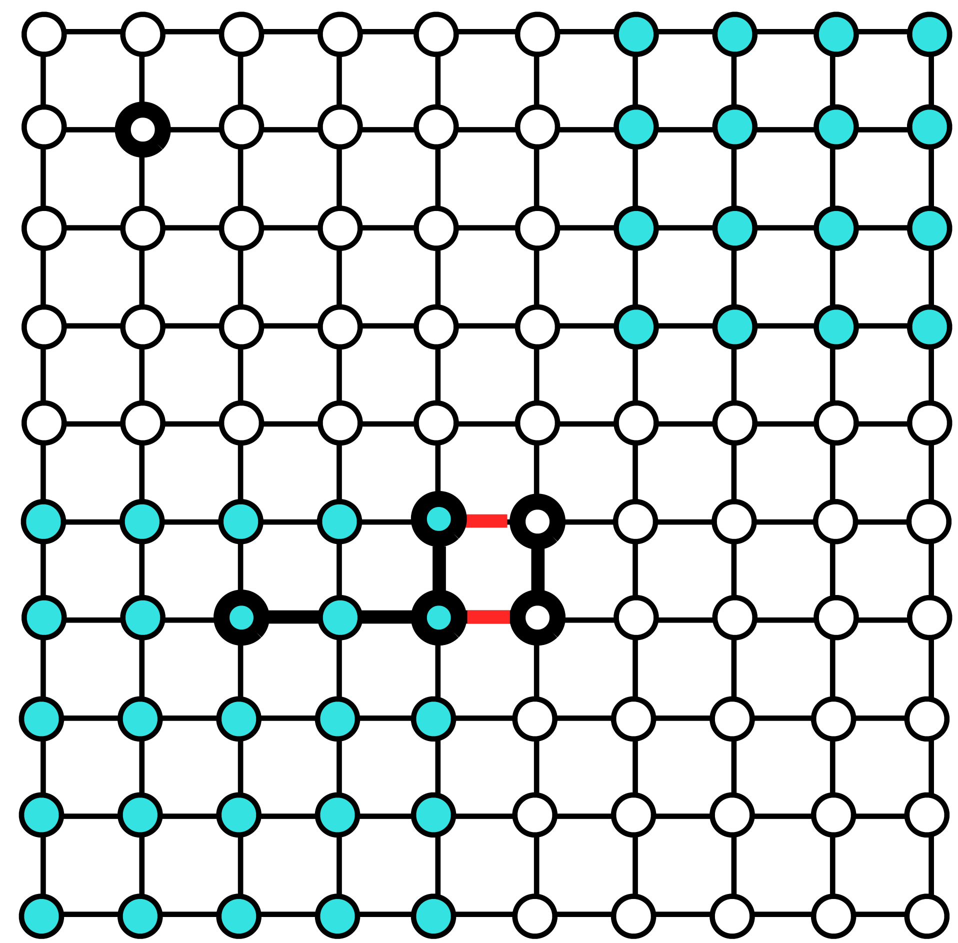
Continue this and remove any cut edges found

The S^2 Algorithm

- Randomly query vertices till you find a pair with opposite labels.

- Among all shortest paths connecting oppositely labeled vertices, pick the shortest one and query the vertex at its mid-point.
  - If you find a "cut" edge, remove it and proceed.

- If there are no more oppositely labeled vertices, go back to step 1 (random sampling).



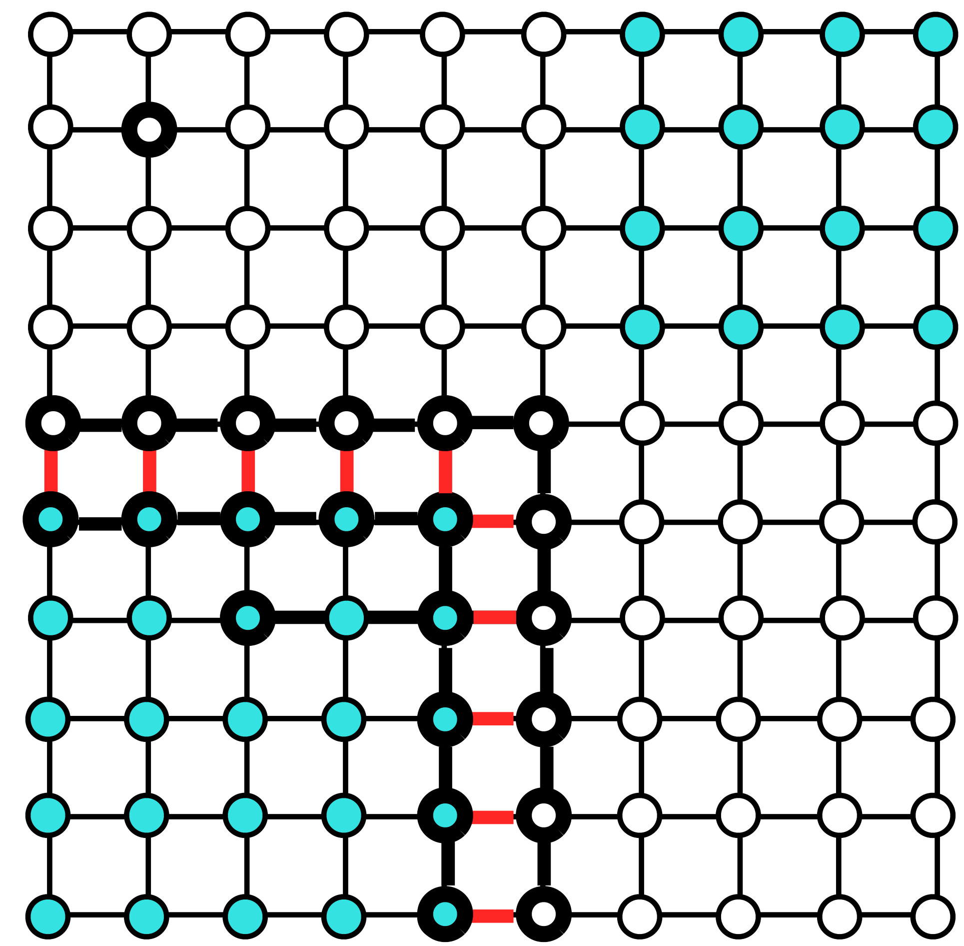
Continue this and remove any cut edges found

The S^2 Algorithm

- Randomly query vertices till you find a pair with opposite labels.

- Among all shortest paths connecting oppositely labeled vertices, pick the shortest one and query the vertex at its mid-point.
  - If you find a "cut" edge, remove it and proceed.

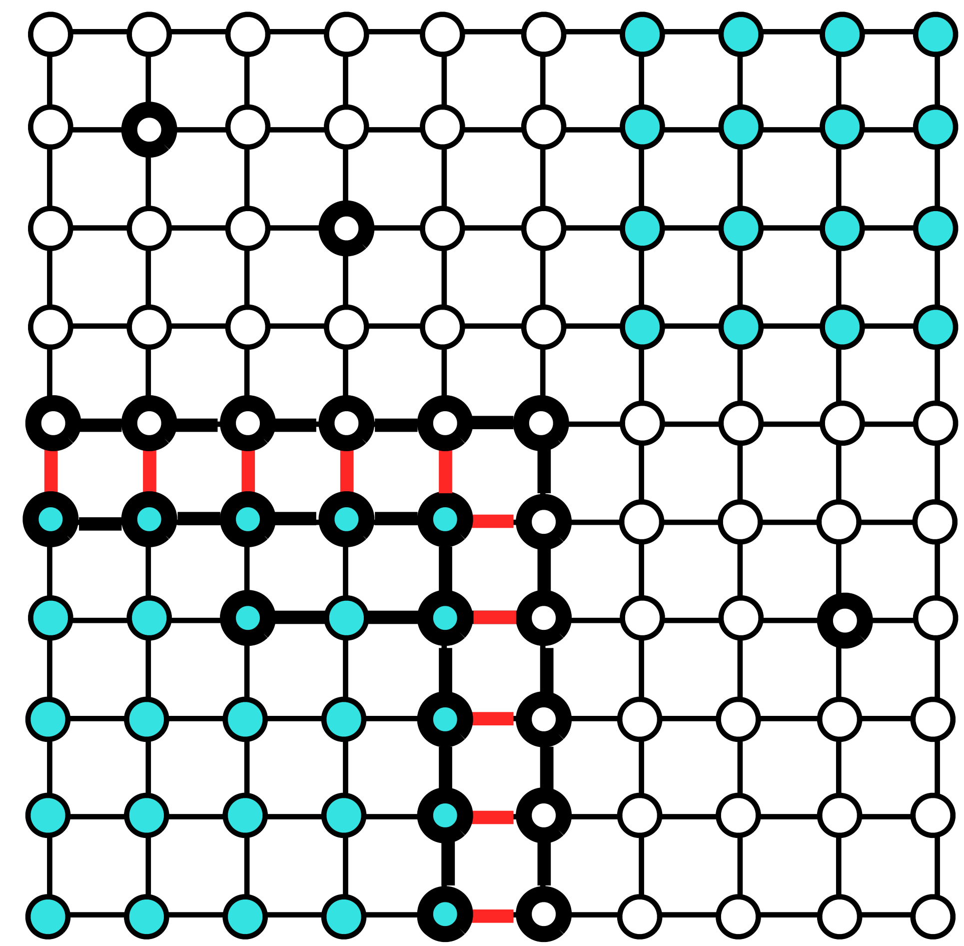
- If there are no more oppositely labeled vertices, go back to step 1 (random sampling).



Continue this and remove any cut edges found

The S^2 Algorithm

- Randomly query vertices till you find a pair with opposite labels.
- Among all shortest paths connecting oppositely labeled vertices, pick the shortest one and query the vertex at its mid-point.
  - If you find a "cut" edge, remove it and proceed.
- If there are no more oppositely labeled vertices, go back to step 1 (random sampling).



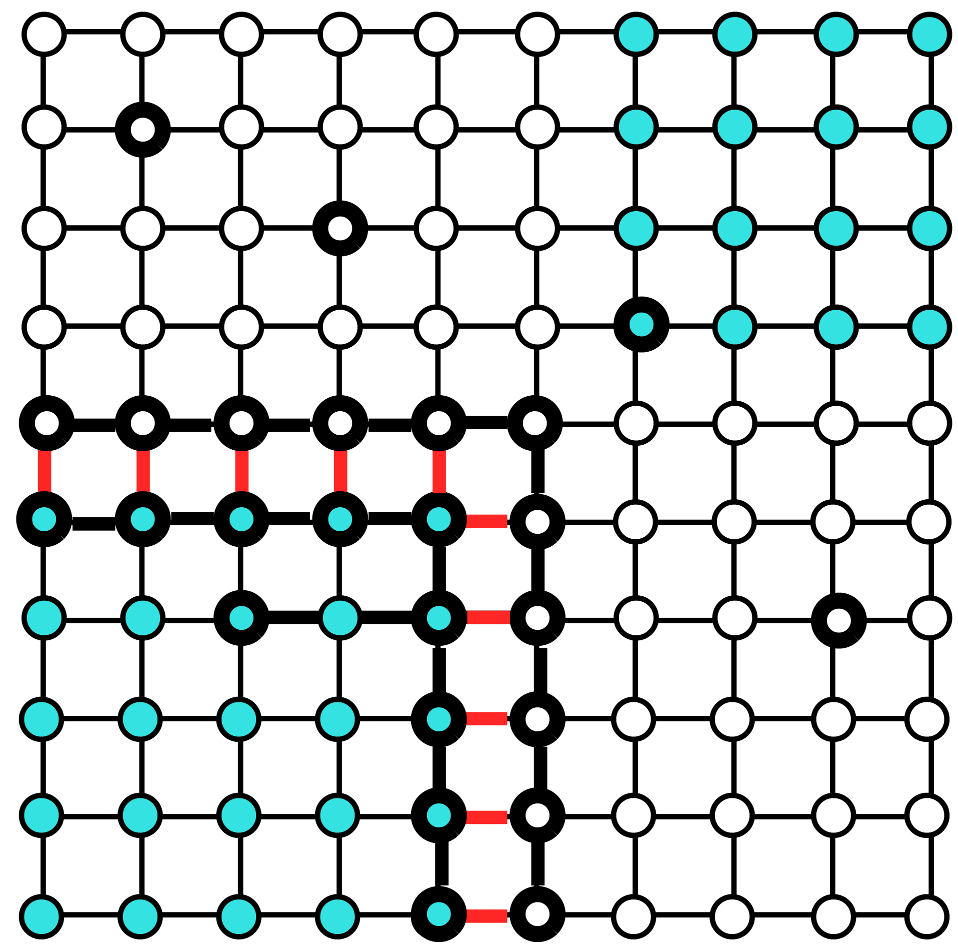
Randomly query again till we find connected pairs of oppositely labeled vertices

The S^2 Algorithm

- Randomly query vertices till you find a pair with opposite labels.

- Among all shortest paths connecting oppositely labeled vertices, pick the shortest one and query the vertex at its mid-point.
  - If you find a "cut" edge, remove it and proceed.

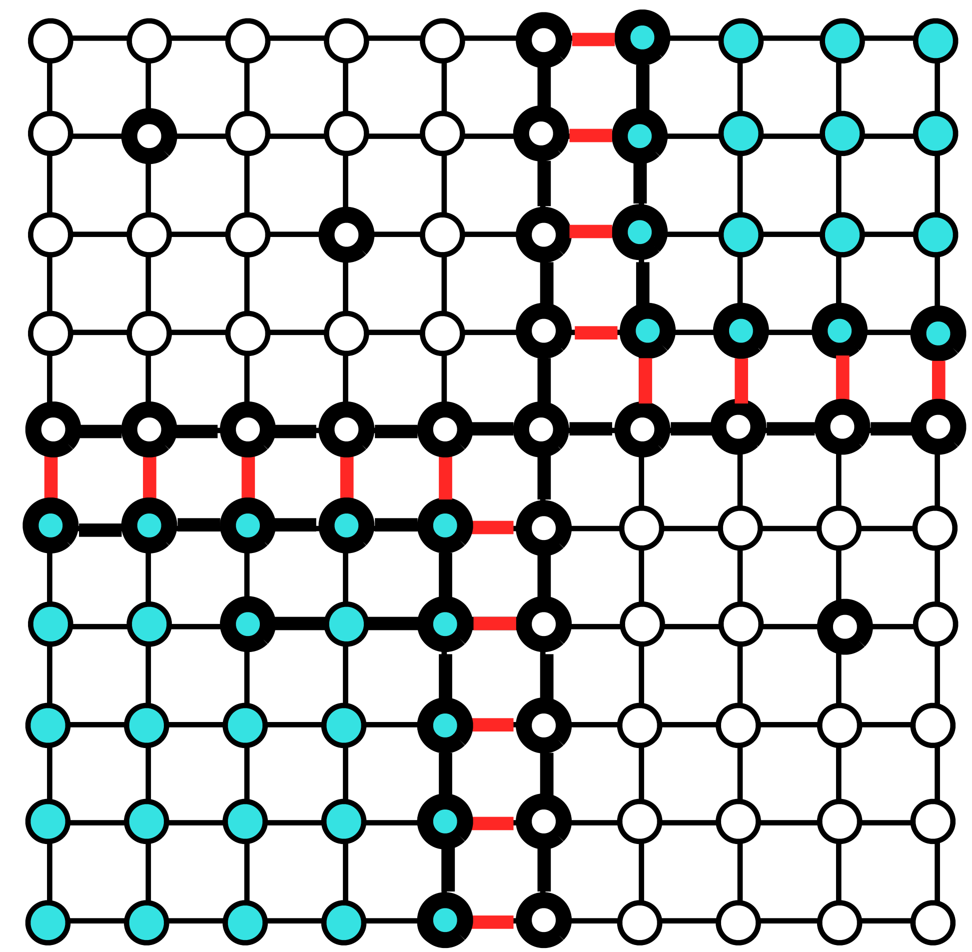
- If there are no more oppositely labeled vertices, go back to step 1 (random sampling).



Randomly query again till we find connected pairs of oppositely labeled vertices

The S^2 Algorithm

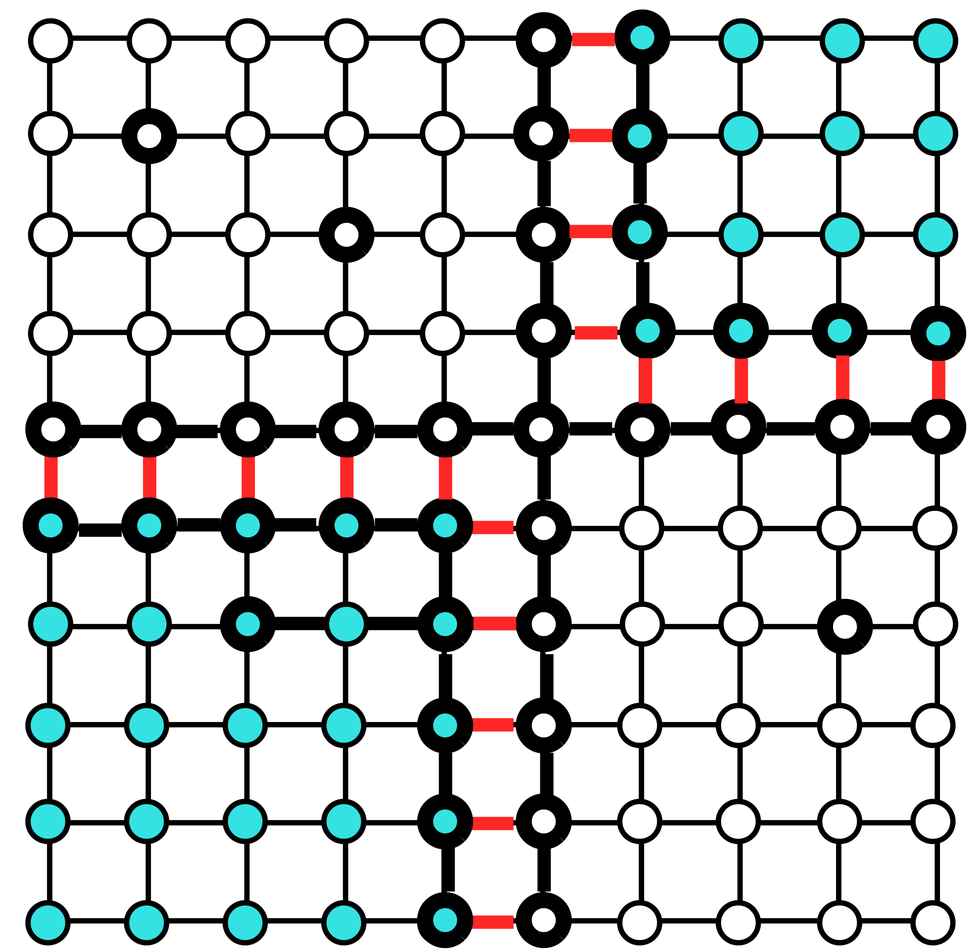
- Randomly query vertices till you find a pair with opposite labels.
- Among all shortest paths connecting oppositely labeled vertices, pick the shortest one and query the vertex at its mid-point.
  - If you find a "cut" edge, remove it and proceed.
- If there are no more oppositely labeled vertices, go back to step 1 (random sampling).



Back to bisecting shortest shortest paths

The S^2 Algorithm

- Randomly query vertices till you find a pair with opposite labels.
- Among all shortest paths connecting oppositely labeled vertices, pick the shortest one and query the vertex at its mid-point.
  - If you find a "cut" edge, remove it and proceed.
- If there are no more oppositely labeled vertices, go back to step 1 (random sampling).



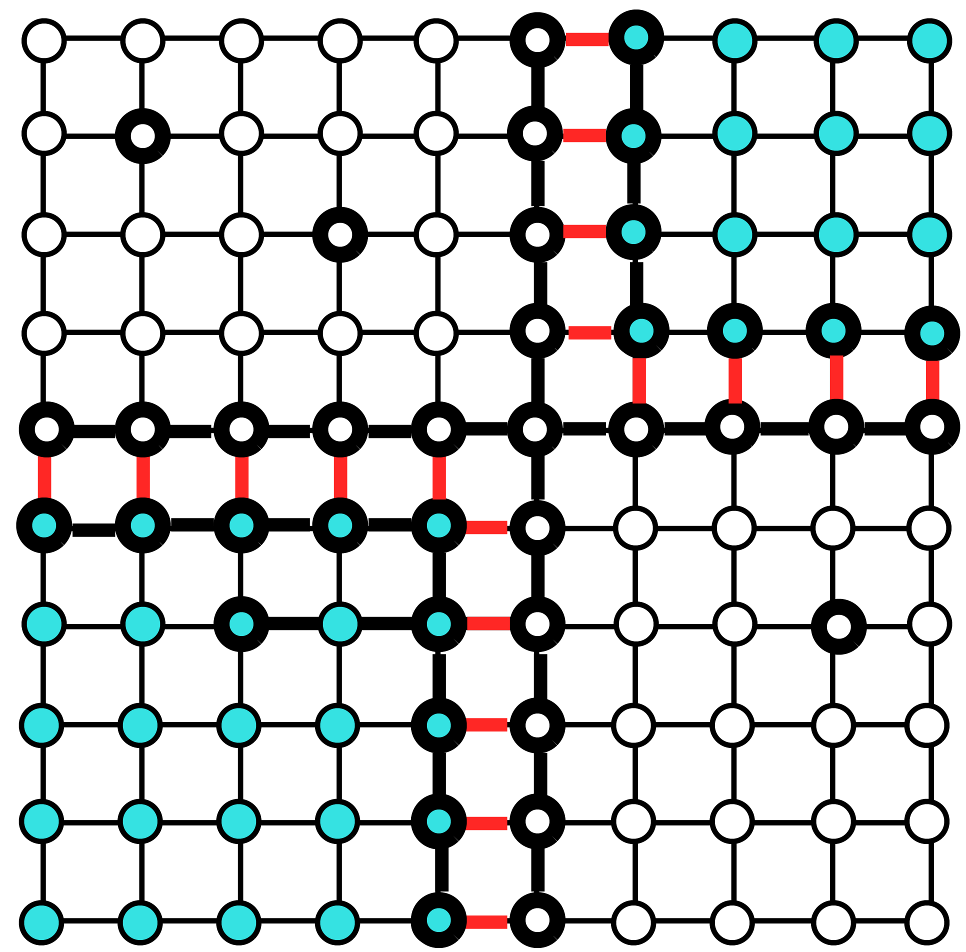
Back to bisecting shortest shortest paths

S^2 thereby very efficiently finds the cut-edges

Visit the poster!

Please visit the poster to:

- Find out about our novel parametrization of the complexity of a binary function with respect to the graph.
  - Size of the cut set (boundary) $|\partial C|$
  - Class balancedness β
  - Cut set composed of m sets of cut edges such that each set is κ clustered.



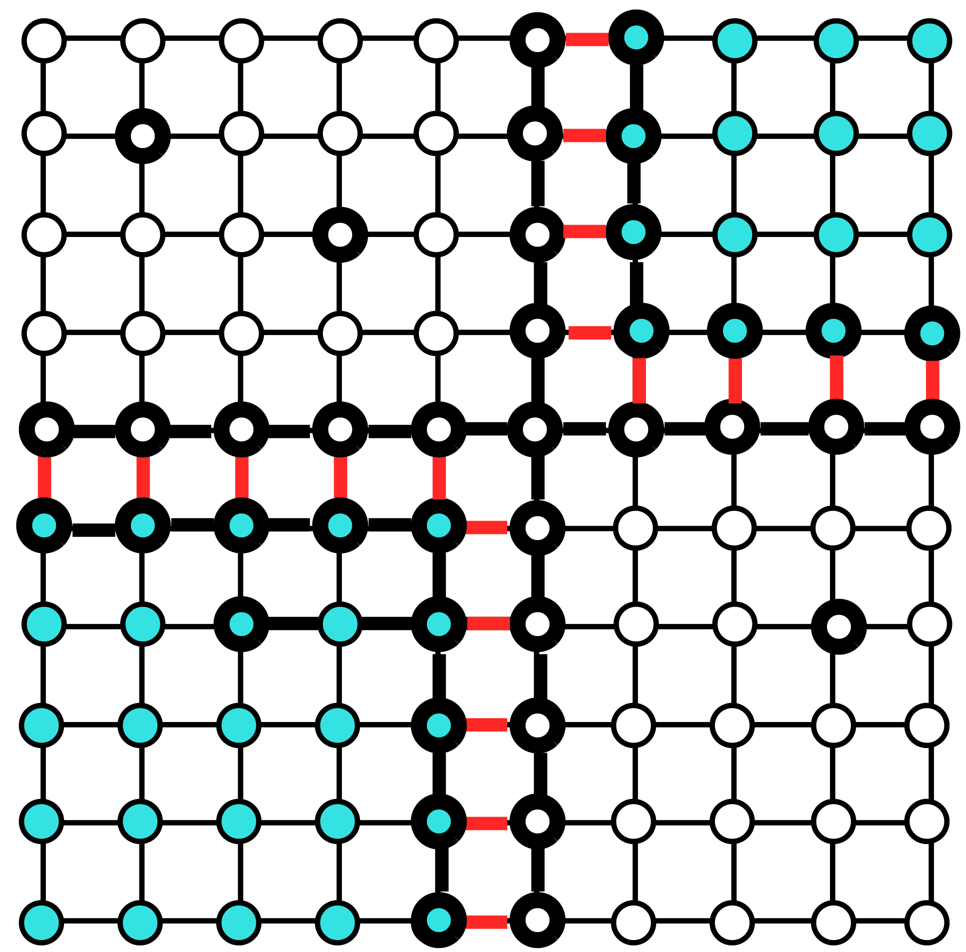
arXiv : 1506.08760

Visit the poster!

Please visit the poster to:

- Find out about our novel parametrization of the complexity of a binary function with respect to the graph.
  - Size of the cut set (boundary) $|\partial C|$
  - Class balancedness β
  - Cut set composed of m sets of cut edges such that each set is κ clustered.
- Find out how to use this to understand the performance of S^2 theoretically.

$$\text{Query complexity} \sim \frac{\log(1/\beta\epsilon)}{\log(1/(1-\beta))} + m\log n + |\partial C|\log\kappa$$

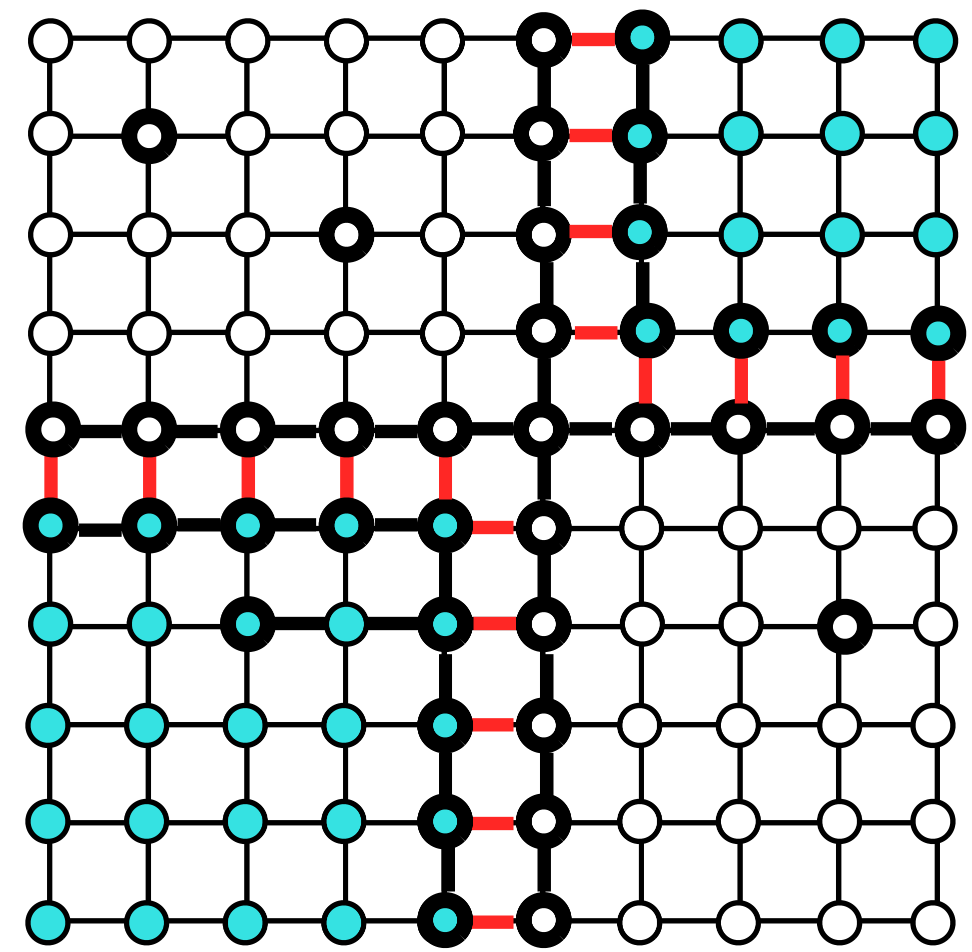


arXiv : 1506.08760

Visit the poster!

Please visit the poster to:

- Find out about our novel parametrization of the complexity of a binary function with respect to the graph.
  - Size of the cut set (boundary) $|\partial C|$
  - Class balancedness β
  - Cut set composed of m sets of cut edges such that each set is κ clustered.



- Find out how to use this to understand the performance of S^2 theoretically.

$$\text{Query complexity} \sim \frac{\log(1/\beta\epsilon)}{\log(1/(1-\beta))} + m \log n + |\partial C| \log \kappa$$

- Learn how S^2 achieves near minimax optimal sample complexity for nonparametric active classification.
  - Consider the lattice graph and run S^2
  - For a broad class of problems (Bayes decision boundary satisfies some regularity condition), sample complexity is

$$\mathcal{O}\left(\left(\frac{\log n}{n}\right)^{1/d-1}\right)$$

arXiv : 1506.08760