# Graph Reductions with Applications to Robotics

A. Baradwaj  S. Dua  J. Hulett

May 23, 2017

# The Problem

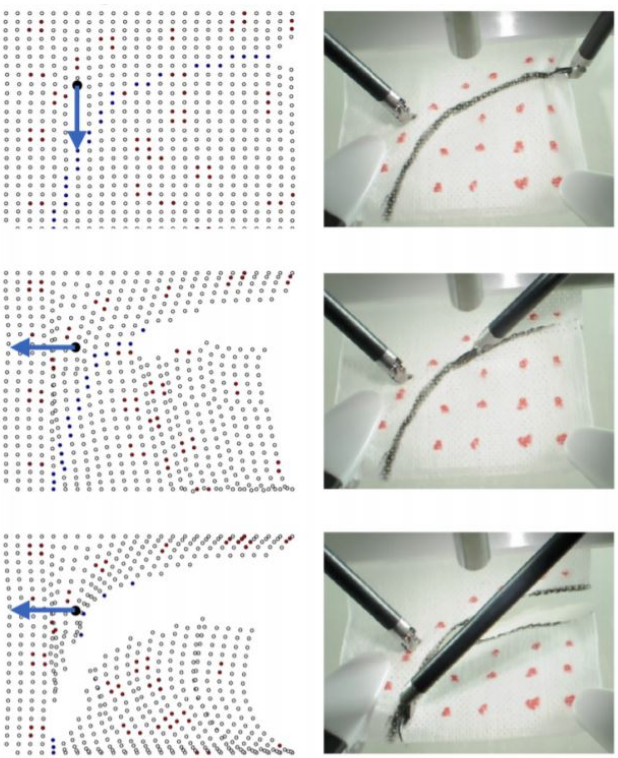

Figure 1: Demonstration of DVRK robot cutting through gauze

# The Problem

- Given a piece of gauze, we'd like to cut it in the most stable way possible
- We model the gauze as a grid of points connected by springs
- We measure a cut's stability by the difference in the potential energy in the system before and after the cut

# Current Methods

- A deep reinforcement learning implementation for this problem exists, but takes far too long. (5 minutes)
- Goal: come up with an approximation that is quick but does not need to be optimal.

# Finding the Potential

"Consider the vector $P = X^T \mathcal{L} X$. $P \in \mathbb{R}^{1\times 3}$, and has the following form:

$$P = [\sum_{(i,j)\in E} (x(i) - x(j))^2 \sum_{(i,j)\in E} (y(i) - y(j))^2 \sum_{(i,j)\in E} (z(i) - z(j))^2]$$

By Hooke's Law, the vector $P$ is the energy in the spring system."

# Finding the Potential

- Not too far off: actually, $P = \mathrm{Tr}(X^T \mathcal{L} X)$
- Consider $(X^T D X)_{(0,0)} = X_0^T D X_0 = \sum_{i \in V} d(i) x(i)^2$
- And $(X^T A X)_{(0,0)} = X_0^T A X_0 = \sum_{(i,j) \in E} 2x(i)x(j)$
- $\sum_{i \in V} d(i) x(i)^2 = \sum_{(i,j) \in E} x(i)^2 + x(j)^2$

# Finding the Potential

- From physics, steady state minimizes potential energy
- So need $X$ that minimizes $\mathrm{Tr}(X^T \mathcal{L} X)$
- Trivial solution when $X = 0$, so must enforce fixed points
- "The solution to the above optimization is a linear system of equations of the form $B(\mathcal{L}X) = Y$."
- "Can be seen from the KKT conditions"

# KKT conditions

- Have objective function $f(X) = \mathrm{Tr}(X^T \mathcal{L} X)$
- $\nabla f(X) = 2\mathcal{L}X$
- Constraints $g_k(X)$ of the form $X_{(i)} = C_i$
- So $\nabla g_k(X)$ has ones in the $i$th row, zeros elsewhere
- Solution has $\nabla f(X) - \sum_k \lambda_k \nabla g_k(X) = 0$ and all constraints satisfied
- Can treat each coordinate separately

# KKT conditions

- Define $B'$ as $B$ with all zero rows removed
- Define $Y'$ to only contain fixed points
- $B'X = Y'$ enforces all constraints
- $\begin{bmatrix} \mathcal{L} & (-B')^T \end{bmatrix} \begin{bmatrix} X_i \\ \lambda \end{bmatrix} = 0$ enforces
  $\nabla f(X) - \sum_k \lambda_k \nabla g_k(X) = 0$
- Solution: $\begin{bmatrix} \mathcal{L} & (-B')^T \\ B' & 0 \end{bmatrix} \begin{bmatrix} X_i \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ Y_i' \end{bmatrix}$

# Finding The Optimal Tension

- Want to maximize $\mathrm{Tr}(X^T \mathcal{L} X) - \mathrm{Tr}(X'^T \mathcal{L}' X')$ over choices of tensioning location
- Use gradient ascent to maximize it
- Gradient is complicated, so use finite difference approximations
- No global maximum–constrain to not move too far from resting position

# Finding The Optimal Tension

- This is faster than the RL method, but still slow
- Can speed up if we make the graph smaller, but need to maintain structure

# Edge Contraction

- Uses a greedy heuristic to contract edges and reduce the size of the graph.
- When 2 nodes are merged, they become connected to all the neighbors of the original 2 nodes.
- So in order to find a clustering, we continuously contract edges.
- Used the QuickUnion data structure (from 61B!) to keep track of clusters.

## Heuristics

- Choose the 2 nodes in the graph with largest (or smallest) degree, find the shortest path between them, and merge them. Continue recursively
  - Works well on barbell graphs and trees.
  - Not so good on a grid graph.
- Choose the node with smallest degree in the graph, merge it with its neighbors
  - Works reasonably well!

# Clusters

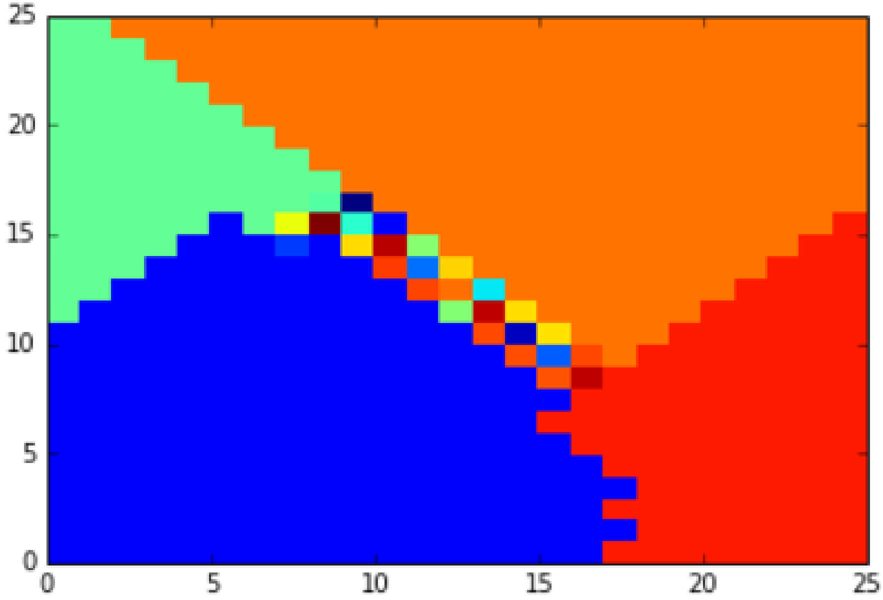


Figure 2: Heuristic: Greedily add smallest degree vertex. k = 30

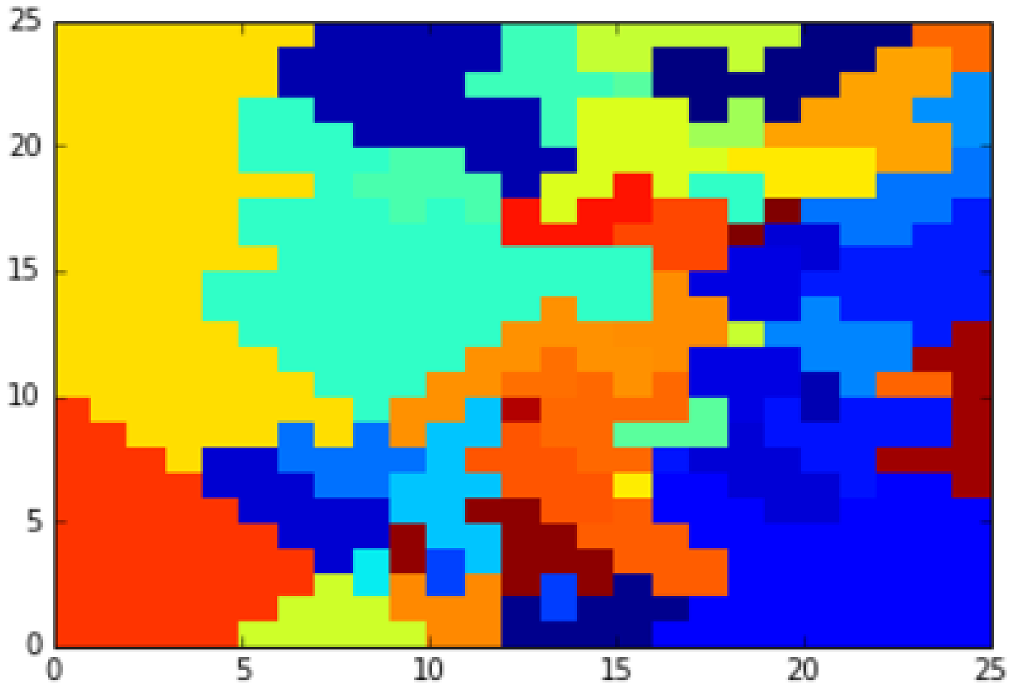


Figure 3: Heuristic: Greedily add smallest degree vertex, prefer to merge with smaller clusters. k = 20

# Spectral Clustering

- Approximation to sparsest cut problem.
- Given $\mathcal{L}$ of the graph, compute the unit eigenvectors corresponding to the $k$ smallest eigenvalues to minimize Rayleigh quotient.
- Cluster these vectors (with k-means)

# Spectral Clustering Visualization

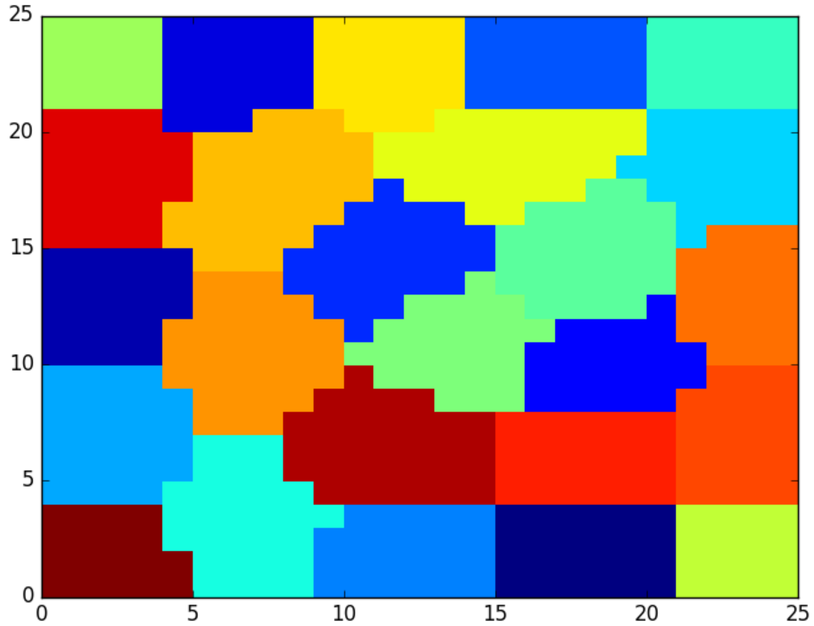


Figure 4: Clustering the grid graph

# Spectral Clustering Speed

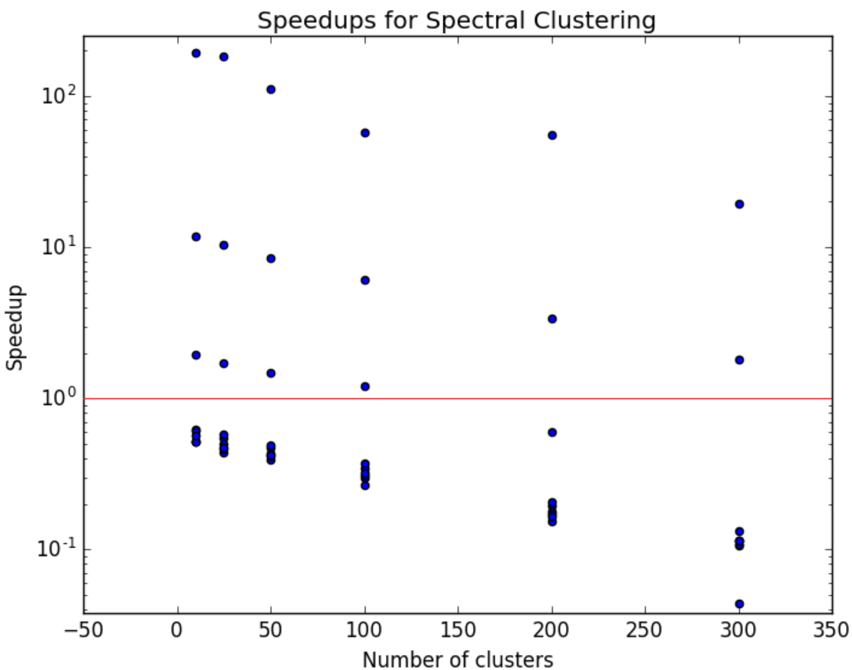


Figure 5: Speedup $= \dfrac{\text{Optimal tension time}}{\text{Approximate tension time}}$
Bigger is better

# Spectral Clustering Performance

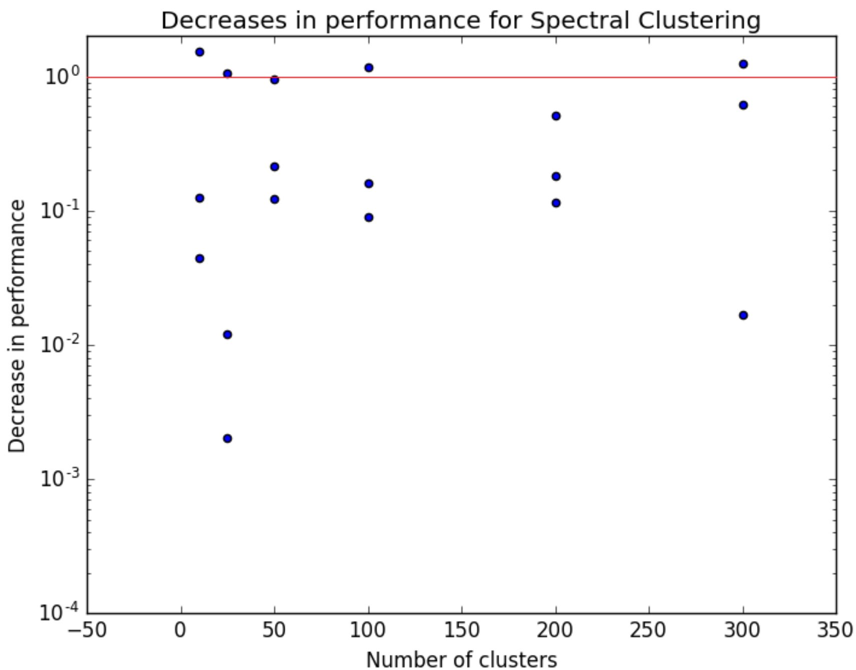


Figure 6: Performance $= \frac{\text{Optimal tension} - \text{Approximate tension}}{\text{Optimal tension} - \text{No tension}}$
Smaller is better

# Our Implementation

`https://github.com/j-m-h/270-project`