Walk Modularity: Graph partitioning based on a generalization of modularity

David Mehrle[1]
Carnegie Mellon University
dmehrle@cmu.edu

Amy Strosser[1]
Mount St. Mary's University
amstrosser@email.msmary.edu

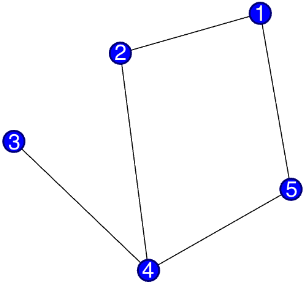
[1] This research was supported by a National Science Foundation Research Experiences for Undergraduates Grant (Award #1062128) hosted by the Rochester Institute of Technology with co-funding from the Department of Defense

Graph Theory Background

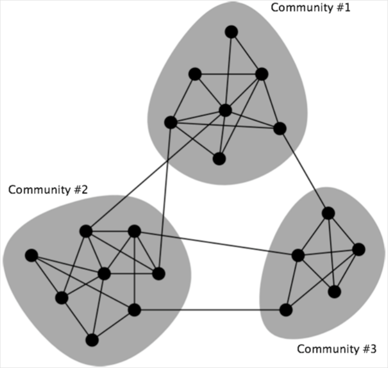
- Consider an undirected graph G with n vertices and m edges
- Adjacency matrix is the $n \times n$ symmetric matrix A with

$$A_{ij} = \begin{cases} 1 & \text{nodes } i \text{ and } j \text{ are connected by an edge} \\ 0 & \text{otherwise} \end{cases}$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$



Modularity



Communities should have more edges within them than the number of edges you would expect based on random chance.

Modularity

Definition: Modularity (Newman, 2004)

$$Q = \frac{1}{2m} \sum_{i,j} (A_{ij} - P_{ij})\, \delta(c_i, c_j)$$

- Compares actual vs. expected number of edges within clusters
- A_{ij} edges actually fall between vertices i and j
- Expect $P_{ij} = \frac{k_i k_j}{2m}$ edges between vertices i and j
- k_i is the degree of vertex i
- c_i is the group to which vertex i belongs

$$\delta(c_i, c_j) = \begin{cases} 1 & c_i = c_j \\ 0 & \text{otherwise} \end{cases}$$

Walk Modularity

Definition: Walk Modularity

$$Q_\ell = \frac{1}{2m_\ell} \sum_{i,j} \left((A^\ell)_{ij} - (P^\ell)_{ij} \right) \delta(c_i, c_j)$$

- Compares actual vs. expected number of walks of length ℓ
- $(A^\ell)_{ij}$ is the number of walks of length ℓ between i and j
- $(P^\ell)_{ij}$ is the expected number of walks of length ℓ between i, j
- m_ℓ is the number of walks of length ℓ in the graph

Walk Partitioning

- Partition the graph into two communities by maximizing Q_ℓ
- Define the partition vector $\mathbf{s}$ by

$$s_i = \begin{cases} +1 & \text{vertex } i \text{ in cluster 1} \\ -1 & \text{vertex } i \text{ in cluster 2} \end{cases}$$

- Let $B_\ell = A^\ell - P^\ell$
- Note $\delta(c_i, c_j) = \frac{1}{2}(1 + s_i s_j)$

$$Q_\ell = \sum_{i,j} \left((A^\ell)_{ij} - (P^\ell)_{ij} \right)(1 + s_i s_j) = \underbrace{\sum_{i,j} (B_\ell)_{ij}}_{\text{constant}} + \underbrace{\mathbf{s}^T B_\ell \, \mathbf{s}}_{\text{maximize}}$$

- There are 2^n possible choices for $\mathbf{s}$, brute force is not practical
- We can find an approximate optimal solution

Maximizing Walk-Modularity

- Expand in terms of orthonormal eigenvectors $\mathbf{u}_i$ of B_ℓ:

$$\mathbf{s} = \sum_{i=1}^{n} a_i \mathbf{u}_i \ , \qquad a_i = \mathbf{u}_i^T \mathbf{s}$$

- To maximize Q_ℓ, concentrate as much weight as possible on largest eigenvalue

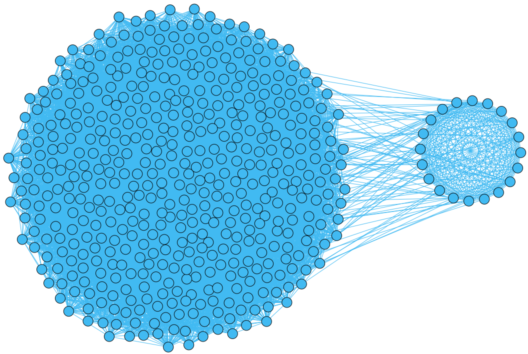
$$Q_\ell = \mathbf{s}^T B_\ell \, \mathbf{s} = \left(\sum_i a_i \mathbf{u}_i^T \right) B_\ell \left(\sum_j a_j \mathbf{u}_j \right) = \sum_{i=1}^{n} (\mathbf{u}_i^T \mathbf{s})^2 \beta_i$$

- If β is largest eigenvalue of $A^\ell - P^\ell$, with eigenvector $\mathbf{u}$, choose $\mathbf{s}$:

$$s_i = \begin{cases} +1 & u_i \geq 0 \\ -1 & u_i < 0 \end{cases}$$

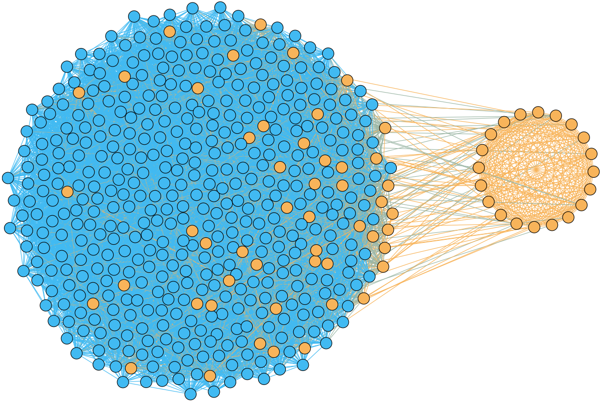
Embedded K_{20}

- Erdős-Rényi random graph on 500 nodes with embedded K_{20}
- Probability of edge between 2 nodes in random graph is 10%
- Probability of edge between node in random graph and node in K_{20} is 5%



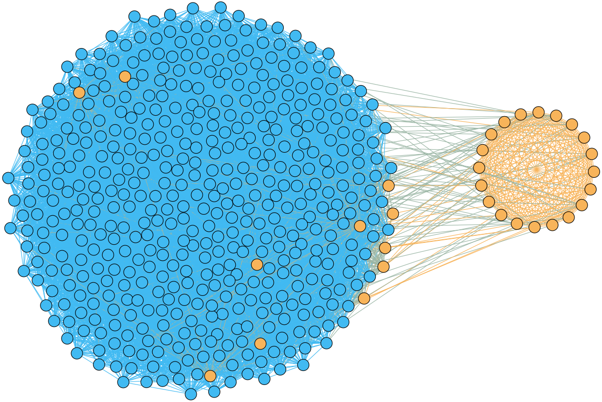
Embedded K_{20}

- Partitioned using $\ell = 1$, regular modularity



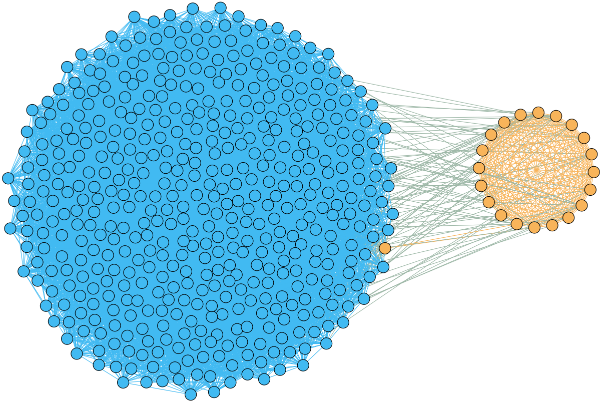
Embedded K_{20}

- Partitioned using $\ell = 2$, walks of length 2



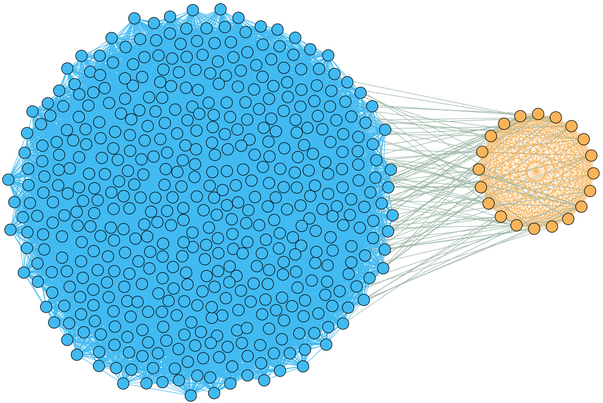
Embedded K_{20}

- Partitioned using $\ell = 3$, walks of length 3



Embedded K_{20}

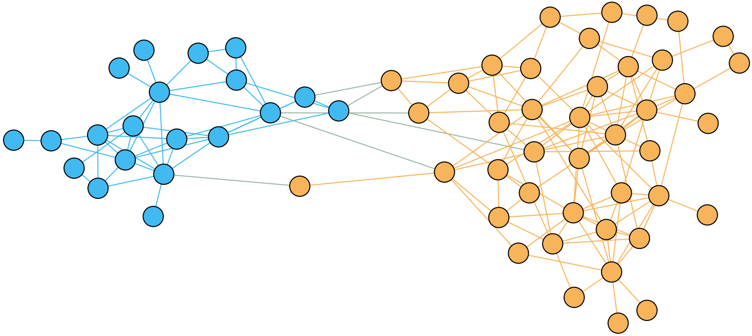
- Partitioned using $\ell = 4$, walks of length 4



Rule of thumb for choosing ℓ: $\ell \approx$ diameter of G

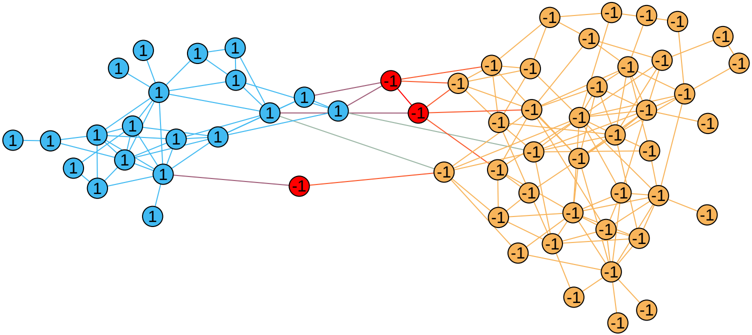
Dolphin Network (Lusseau 2003)

- A group of 62 dolphins were tracked over ten years
- The group split in two after one of the dolphins departed
- A standard test used in literature for graph partitioning algorithms



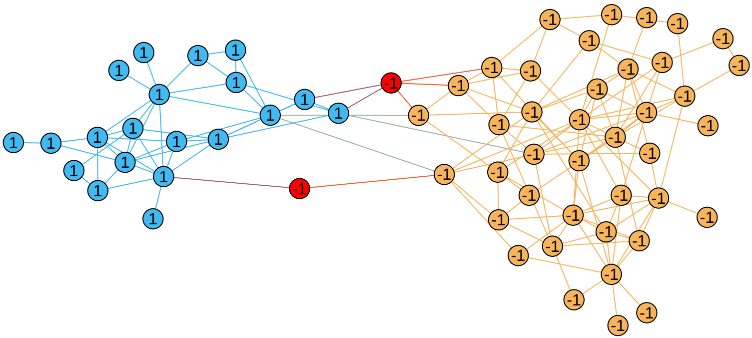
Dolphin Network

- Modularity partition, $\ell = 1$
- ± 1 indicates the observed partitioning of the dolphin network
- Red nodes are incorrectly placed relative to observed



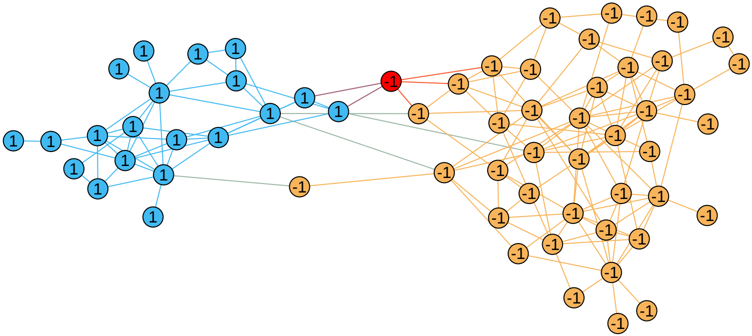
Dolphin Network

- Q_8 walk-modularity partition, walks of length 8
- ± 1 indicates the observed partitioning of the dolphin network
- Red nodes are incorrectly placed relative to observed



Dolphin Network

- Q_{10} walk-modularity partition, walks of length 10
- ± 1 indicates the observed partitioning of the dolphin network
- Red nodes are incorrectly placed relative to observed



Multiple Communities

- Recursively divide each community with spectral methods
- For each subdivision, consider change in walk-modularity

$$\Delta Q_\ell = \underbrace{Q_{\ell\,final}}_{\text{after subdivide}} - \underbrace{Q_{\ell\,initial}}_{\text{before subdivide}}$$

- If splitting up a community gives $\Delta Q_\ell < 0$, don't subdivide
- If all nodes are in single community, don't subdivide

Benchmark Tests (Lancichinetti et al. 2008)

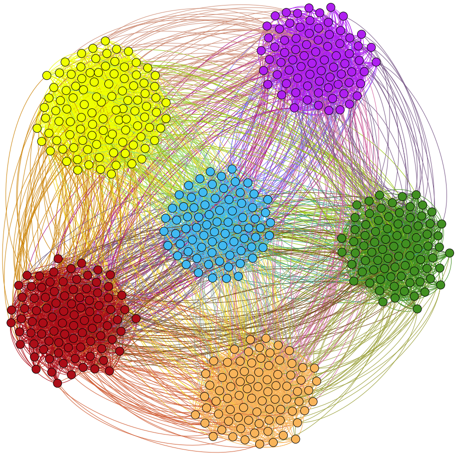
- Benchmark test for community detection algorithms designed by Lancichinetti et. al. 2008
- Joins communities based on a mixing parameter, μ
  - Moves edges between communities with probability μ
- The following slides have a community generated with

$$n = 500, \quad \mu = 0.15, \quad \bar{k} = 25$$

- Each vertex is placed within a single well-defined community

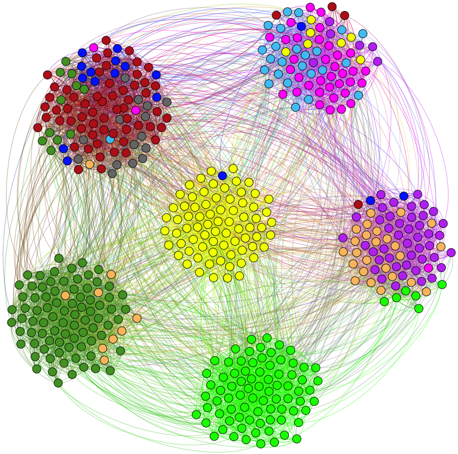
Benchmark Test

- The communities as defined by the test generator



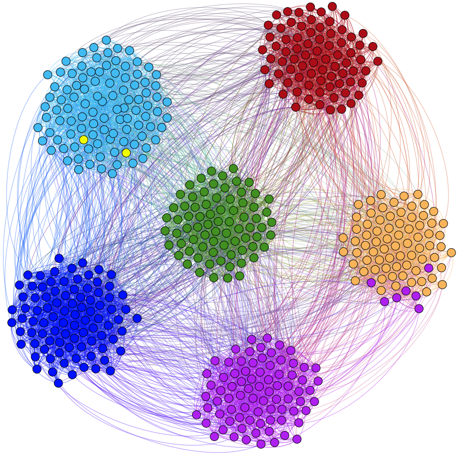
Benchmark Test (Modularity)

- The communities as found by edge-modularity ($\ell = 1$)



Benchmark Test ($\ell = 8$)

- The communities as found by walk-modularity ($\ell = 8$)



Computational Complexity

- Same asymptotic complexity as modularity, $O(n^2)$
  - Power method to find leading eigenvector of B_ℓ

$$\mathbf{x}_{n+1} = \frac{B_\ell\,\mathbf{x_n}}{\|B_\ell\,\mathbf{x}_n\|}\ , \qquad \mathbf{x}_1 \in \mathbb{R}^n \text{ random}$$

  - Repeated multiplication against vector avoids computing matrix powers

$$(A^\ell - P^\ell)\mathbf{x} = \underbrace{A \cdot A \cdot A \cdots A}_{\ell \text{ times } ,O(n^2)}\mathbf{x} - \underbrace{P \cdot P \cdot P \cdots P}_{\ell \text{ times } ,O(n^2)}\mathbf{x}$$

- Comparably fast in practice as well, above tests < 1 s for most ℓ

Conclusions

- In most of our real-world and benchmark tests so far, walk-modularity performs significantly better than edge-modularity
- Comparable speed both asymptotically and practically
- Very similar to modularity, which is often used in practice

Acknowledgements

Thank you to . . .

- Dr. Anthony Harkin, for mentoring and suggestions
- Dr Darren Narayan, for organizing the REU
- Rochester Institute of Technology
- the National Science Foundation, for grant #1062128
- the Department of Defense, for co-funding
- The AMS and MAA for organizing the JMM