Similarity Metrics in Networks

Mentee: Samuel Hsu, Mentor: Vydhourie Thiyageswaran

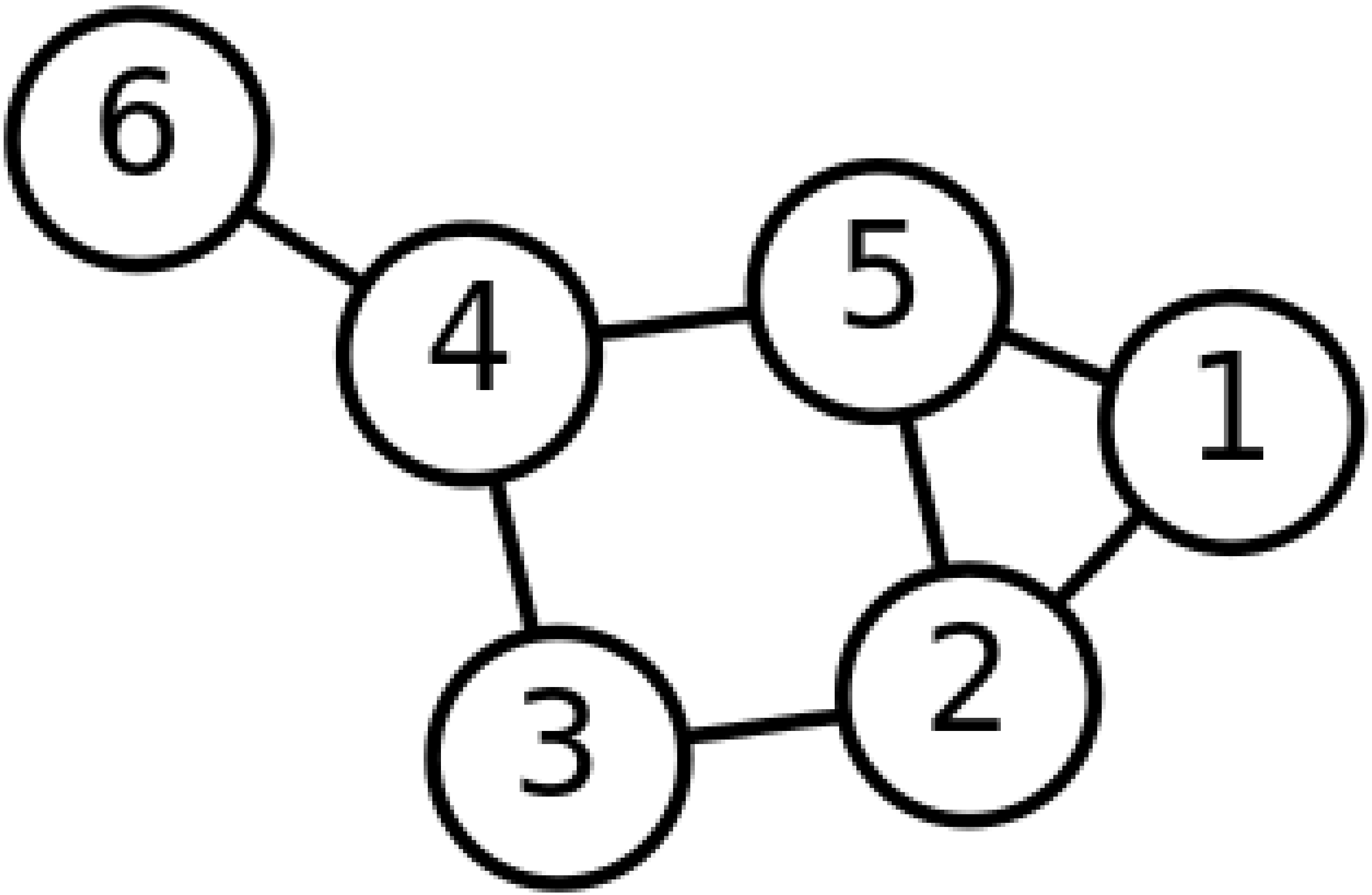
Table of Contents

Basics of Graphs and Associated Matrices

Graphs

- Encyclopaedia Britannica: A **graph** is a network of points connected by lines
- The points in a graph are **vertices** or **nodes** while the lines are **edges**
- Wikipedia: A graph is a mathematical structure used to model pairwise relations between objects.
- Spielman: Graphs are used to model *connections* between things
- Graph edges can sometimes have weights and directions
- Every vertex has a degree: in an unweighted graph, it’s the number of edges connected to a vertex, and in a weighted graph, it’s the sum of all of the edge weights connected to a vertex
- Examples of graphs include friendship graphs, network graphs, and circuit graphs

Graphs



User:AzaToth, Public domain, via Wikimedia Commons, https://commons.wikimedia.org/wiki/File:6n-graf.svg

Goal:

In some network or graph of “things”, find some way to calculate the similarity between different “things” and group “things” together.

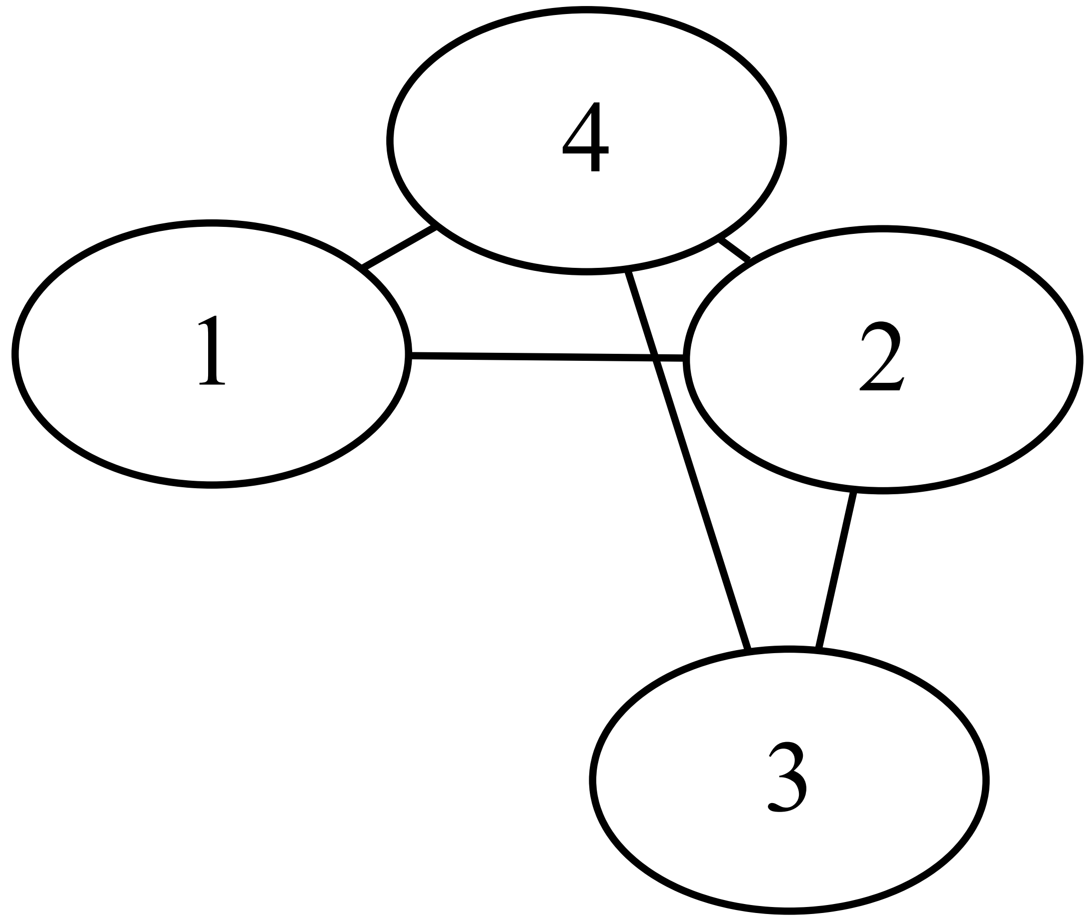
Matrices Associated with Graphs

Common matrices associated with graphs include:

- The adjacency matrix ($\mathbf{A}$)
- The degree matrix ($\mathbf{D}$)
- The graph Laplacian matrix ($\mathbf{L} = \mathbf{D} - \mathbf{A}$)

Don't worry if you don't know matrices or linear algebra (take Math 208 to learn about them); just treat matrices as "tables of numbers" for now.

Example



Example (continued)

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

$$\mathbf{D} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix}$$

Example (continued)

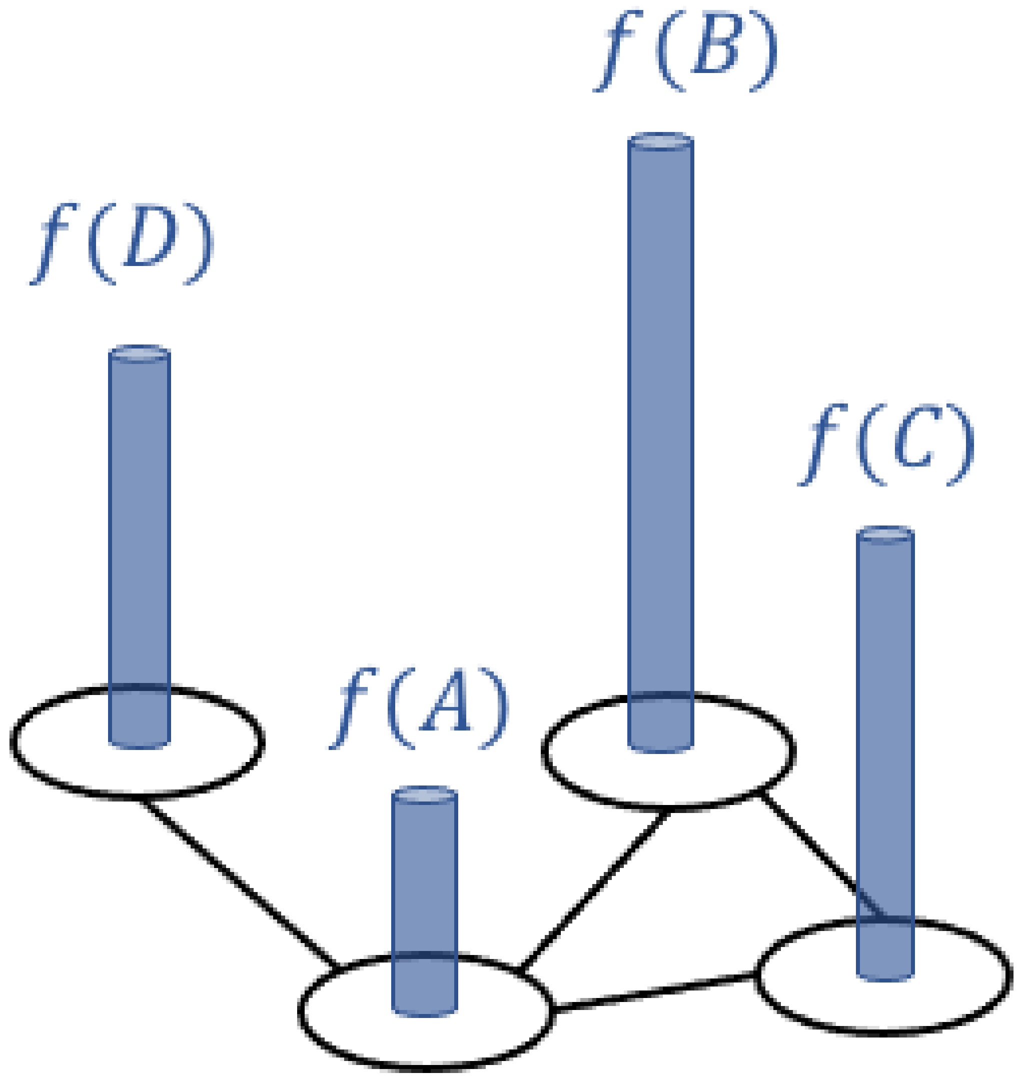
$$\mathbf{L} = \mathbf{D} - \mathbf{A} = \begin{pmatrix} 2 & -1 & 0 & -1 \\ -1 & 3 & -1 & -1 \\ 0 & -1 & 2 & -1 \\ -1 & -1 & -1 & 3 \end{pmatrix}$$

Graph Laplacian

- A *graph function* maps each vertex to a number.
- The graph Laplacian helps measure the "smoothness" of a graph function
  - a graph function is *smooth* if the function doesn't jump too dramatically between connected vertices
- The *smoothness* of a function is given by $\mathbf{f}^T\mathbf{L}\mathbf{f}$ where $\mathbf{f}$ is a column vector representing the value of our graph function at every vertex
- This is equivalent to $\sum_{u\sim v} w_{uv}(f(u) - f(v))^2$
- Smooth functions should minimize this expression

From Daniel Spielman, Muni Sreenivas Pydi, and Matthew Bernstein

Graph Laplacian



From Matthew N. Bernstein at https://mbernste.github.io/posts/laplacian_matrix/

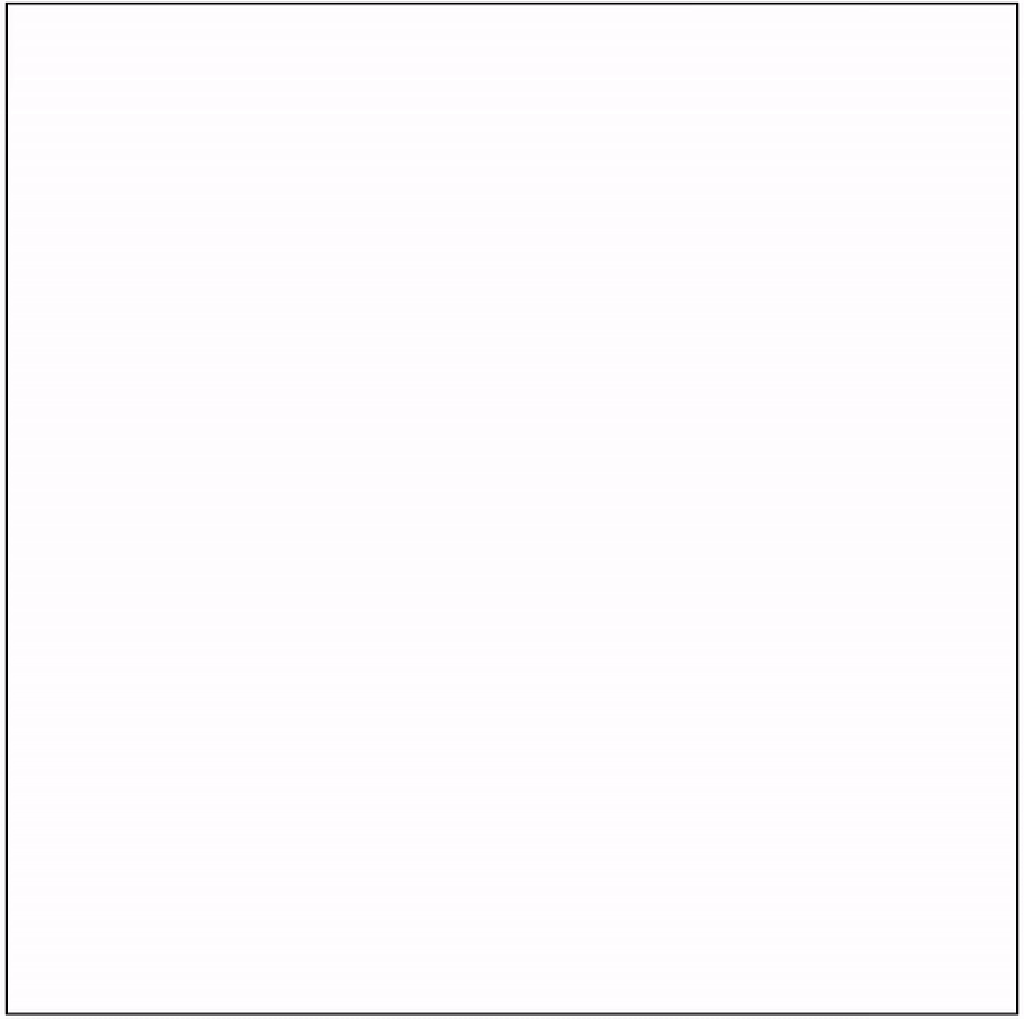
Random Walks on Graphs

Random Walks

A *random walk* on a graph can be understood as follows:

- Imagine you are “standing” at a vertex of a graph
- The next moment, you decide to randomly walk to another vertex
- You repeat this random process a few times.
- The path you take is a random walk.
- In an unweighted graph, you have an equal chance of walking along each edge
- In a weighted graph, you don’t; more strongly-weighted edges are more likely to be walked along
- More similar vertices are connected by more high-weight edges
- Less similar vertices are connected by fewer edges that are lower in weight

Random Walks



Random Walks

- The sequence of nodes visited by a random walker is a random walk
- Random variable $s(t)$ contains the current location of walker
  - $s(t) = i$ means that a walker is at position i at time t
- The probability that the walker visits a neighboring node j at time $t + 1$ given that they were just at node i at time t is $P(s(t+1) = j | s(t) = i)$

(I’ve left out most of the information on Markov Chains for simplicity’s sake; these are helpful for a more complete description of random walks)

Similarity Metrics

Overview

Metrics for computing the similarity between two vertices include:

- the average first passage time $m(k|i)$
- the average first passage cost $o(k|i)$
- the pseudoinverse of the graph Laplacian $\mathbf{L}^+$
- the average commute time $n(i,j)$
- the Euclidean Commute Time Distance $[n(i,j)]^{\frac{1}{2}}$

Goal:

In some network of “things”, find some way to calculate the similarity between “things” and group “things” together

Example: Imagine some database of people and some movies they’ve watched recently.

- “Computing similarities between people allows us to cluster them into groups with similar interest about watched movies.”
- “Computing similarities between people and movies allows us to suggest movies to watch or not to watch.”
- “Computing similarities between people and movie categories allows us to attach a most relevant category to each person.”

From Fouss et al.

Pseudoinverse of the Laplacian

- Not all matrices are invertible (including the Graph Laplacian)
- The Moore-Penrose pseudoinverse generalizes the idea of an inverse matrix
- The pseudoinverse of the Laplacian is calculated as follows:
  - $\mathbf{L}^+ = \left(\mathbf{L} - \frac{\mathbf{e}\mathbf{e}^T}{n}\right)^{-1} + \frac{\mathbf{e}\mathbf{e}^T}{n}$
  - $\mathbf{e}$ is the all-ones vector, and n is the number of nodes
- It's a similarity matrix (the similarity of two vertices i and j can be found by looking at the ith row and jth column of L^+)
- It's also used to calculate some of the remaining quantities

Average First-Passage Time

- The average first passage time is $m(k|i)$
- Defined as the average number of steps that a random walker at node i takes to visit node k
- One way to think about it: $m(k|i) = E[T_{ik}|s(0) = i]$
  - The *expected value* of the minimum time of hitting state *k* if you start at state *i*
- Defined using these formulas:
  - $\begin{cases} m(k|k) = 0 \\ m(k|i) = 1 + \sum_{j=1}^{n} p_{ij} m(k|j) \end{cases}$
  - $m(k|i) = \sum_{j=1}^{n} \left(l_{ij}^{+} - l_{ik}^{+} - l_{kj}^{+} + l_{kk}^{+} \right) d_{jj}$
    - Computed from the pseudoinverse of the Laplacian

Average First-Passage Cost

- The average first passage cost is $o(k|i)$
- Say a random walker incurs a cost $c(j|i)$ if they walk from i to some neighboring vertex j
- Defined as the average cost a random walker incurs if they want to visit any node k from node i
- Defined using these formulas:
  - $\begin{cases} o(k|k) = 0 \\ o(k|i) = \sum_{j=1}^{n} p_{ij} c(j|i) + \sum_{j=1}^{n} p_{ij} o(k|j) \end{cases}$
  - $o(k|i) = \sum_{j=1}^{n} \left(l_{ij}^{+} - l_{ik}^{+} - l_{kj}^{+} + l_{kk}^{+} \right) b_j$
    - Computed from the pseudoinverse of the Laplacian

Average Commute Time

- “Symmetric” version of the average first-passage time
  - $n(i,j) = m(i|j) + m(j|i)$
  - Sum of the average-first passage times *in both directions* between i and j
- Calculated using the following formulas:
  - $n(i,j) = m(i|j) + m(j|i)$
  - $n(i,j) = V_G \left(l_{ii}^{+} + l_{jj}^{+} - 2 l_{ij}^{+} \right)$
    - l_{ab}^{+} is an element of the matrix $\mathbf{L}^{+}$
    - V_G, the volume of the graph, is the sum of all of the degrees
  - $n(i,j) = V_G (\mathbf{e}_i - \mathbf{e}_j)^T \mathbf{L}^{+} (\mathbf{e}_i - \mathbf{e}_j)$
    - $\mathbf{e}_i$ is a standard basis vector (like $< 1, 0, \ldots, 0 >$ or $< 0, 1, \ldots, 0 >$)

Average Commute Time

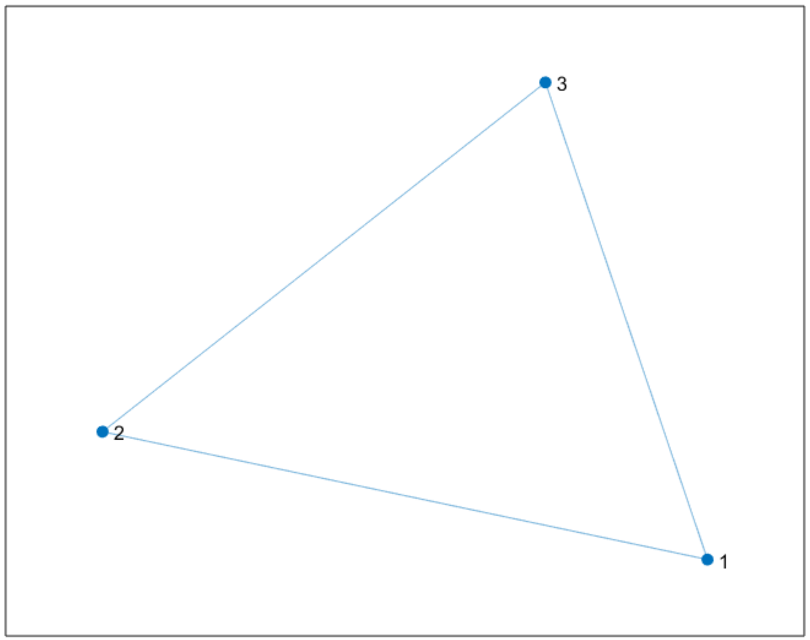
- Interesting tidbit about the average commute time:
- We can treat graphs like networks of *electrical resistors*
  - (Resistors resist the flow of electrical current)
- The average commute time is proportional to the *effective resistance* between the two vertices of the corresponding resistor network
- The weight of each edge is the *inverse* of the resistance
- $r_{ij} = w_{ij}^{-1}$
- Low resistance = high weight, lots of electrical current flows through
- Average commute time is also known as the “commute-time distance” or the “resistance distance”

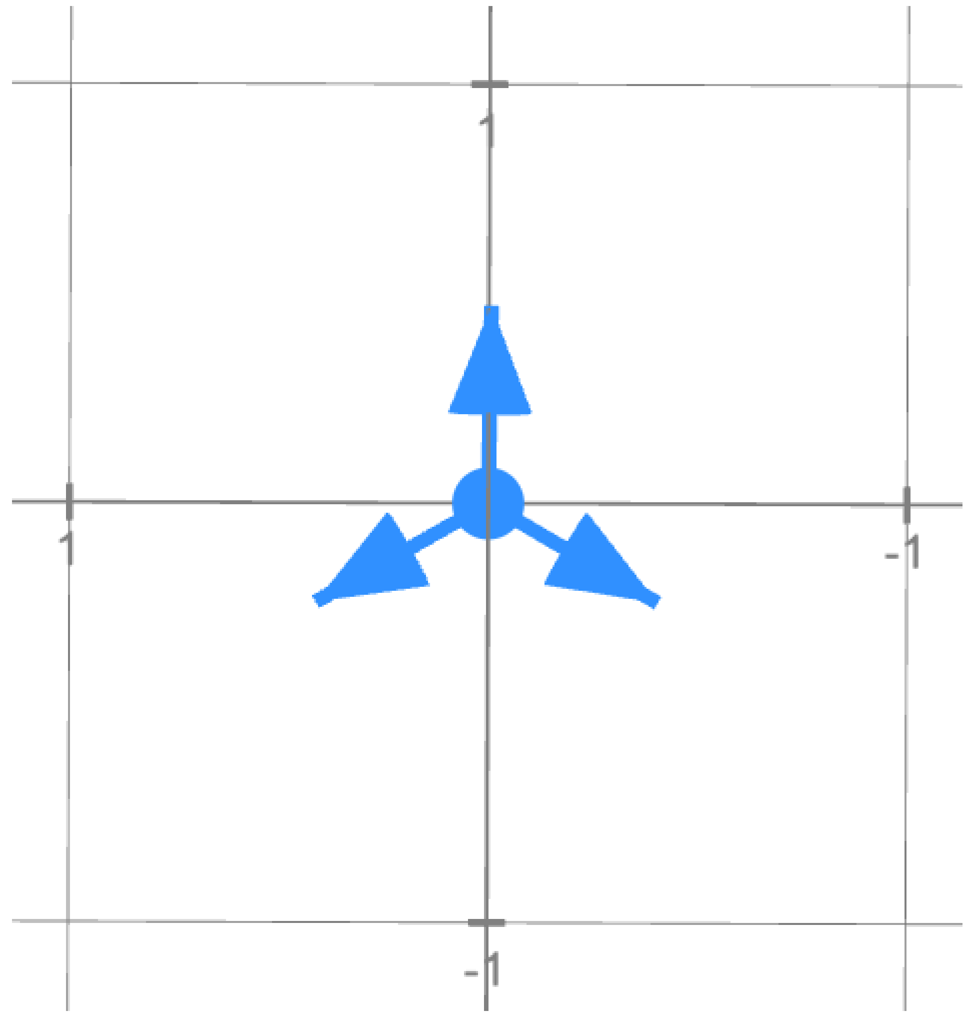
Euclidean Commute Time Distance

- The Euclidean Commute Time Distance is defined as $[n(i,j)]^{\frac{1}{2}}$
- Here's what's interesting:
  - You can define vectors that correspond to each node called *transformed node vectors* using this formula: $\mathbf{x}'_i = \mathbf{\Lambda}^{\frac{1}{2}}\mathbf{U}\mathbf{e}_i$
    - $\mathbf{U}$ contains the eigenvectors of $\mathbf{L}^+$ while $\mathbf{\Lambda}$ is a diagonal matrix with the eigenvalues
    - Procedure for obtaining node vectors comes from the *spectral decomposition* of the pseudoinverse of the Laplacian
  - The distance between the transformed node vectors is exactly the Euclidean Commute Time Distance
- *Inner products* of the node vectors give you the elements of the pseudoinverse of the Laplacian matrix.
- Principal Component Analysis gives you lower-dimensional transformed node vectors that are still roughly separated by the Euclidean Commute Time Distance

Node Vectors of a Graph

The node vectors here appear to be equidistant from one another; this lines up with the observations that all of the edges have the same weight, and all nodes are equidistant.





Left: From MATLAB, Right: From Math3D

Recap and Conclusion

Graphs and Matrices

- A graph is used to model connections between things and is depicted as a network of vertices connected by edges
- Matrices associated with the graph include the adjacency, degree, and graph Laplacian matrix
- The Graph Laplacian measures the smoothness of a function over a graph

Random Walks

- If you stand on a vertex and start randomly walking to nearby vertices, the path you take is a random walk
- Edges with greater weights have a greater probability of being walked to
- Random variable $s(t)$ contains the current location of walker
  - $s(t) = i$ means that a walker is at position i at time t
- The probability that the walker visits a neighboring node j at time $t + 1$ given that they were just at node i at time t is $P(s(t+1) = j | s(t) = i)$

Similarity Metrics

Metrics for computing the similarity between two vertices include:

- the average first passage time $m(k|i)$
- the average first passage cost $o(k|i)$
- the pseudoinverse of the graph Laplacian
  - used to calculate the remaining quantities
  - the best similarity metric
- the average commute time $n(i, j)$
  - proportional to effective resistance
- the Euclidean Commute Time Distance $[n(i, j)]^{\frac{1}{2}}$
  - transformed node vectors are separated by this distance

Similarity Metrics

The goal of these similarity metrics is to group “things” together.

Recall the movie example: We now can calculate metrics that let us recommend movies to users and group similar users together

Sources:

- Spectral and Algebraic Graph Theory by Daniel Spielman
- Random-Walk Computation of Similarities between Nodes of a Graph with Application to Collaborative Recommendation by Fouss et al.
- Quora post on the Graph Laplacian by Muni Sreenivas Pydi
- Cross Validated post on Principal Component Analysis by amoeba
- Post on the Graph Laplacian by Matthew Bernstein
- Linear Algebra with Applications by Jeffrey Holt
- Basics of Applied Stochastic Processes by Richard Serfozo
- OpenStax University Physics, Volume 2
- Optimization Models by Laurent El Ghaoui