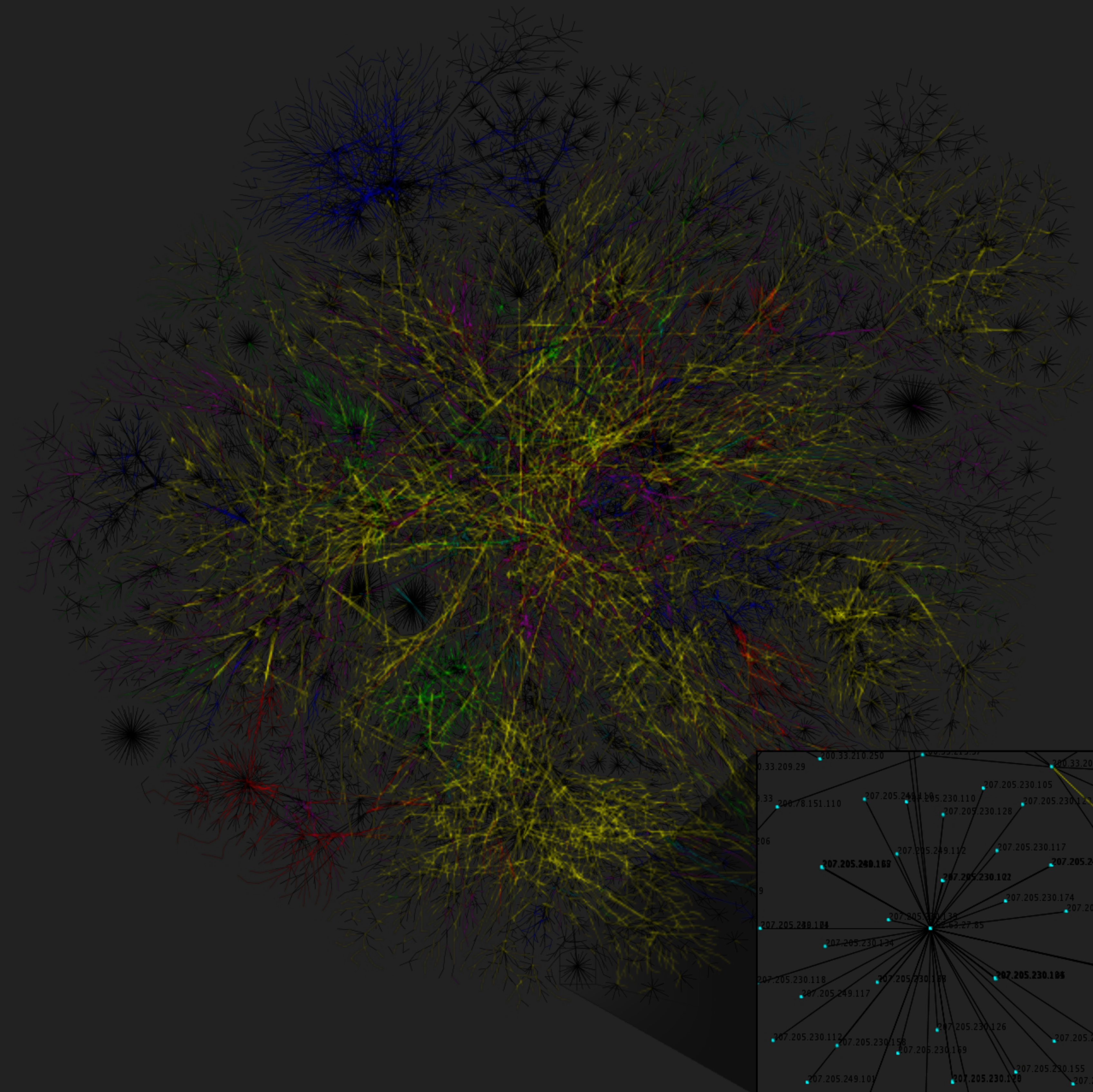
CSCI 5922 - NEURAL NETWORKS AND DEEP LEARNING

GRAPHS AND GRAPH CONVOLUTIONAL NETWORKS

GRAPHS ARE COMMON IN NATURE AND THE HUMAN-MADE WORLD

A partial graph of the Internet circa 2005. Path length denotes latency. Color denotes top-level domain (TLD).

Explaining real-world graphs requires understanding their individual parts and components, combined with domain knowledge.

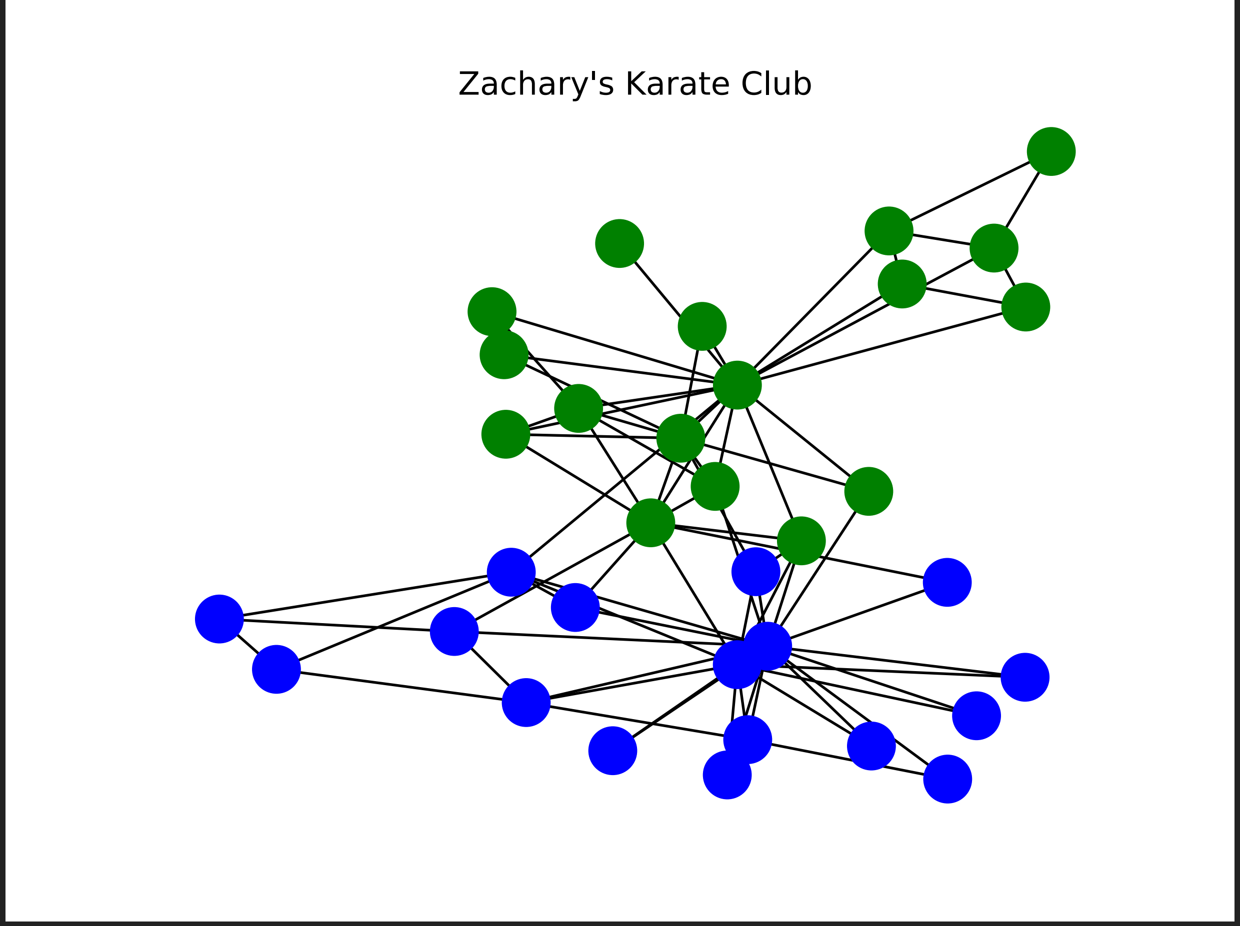


https://en.wikipedia.org/wiki/History_of_the_Internet

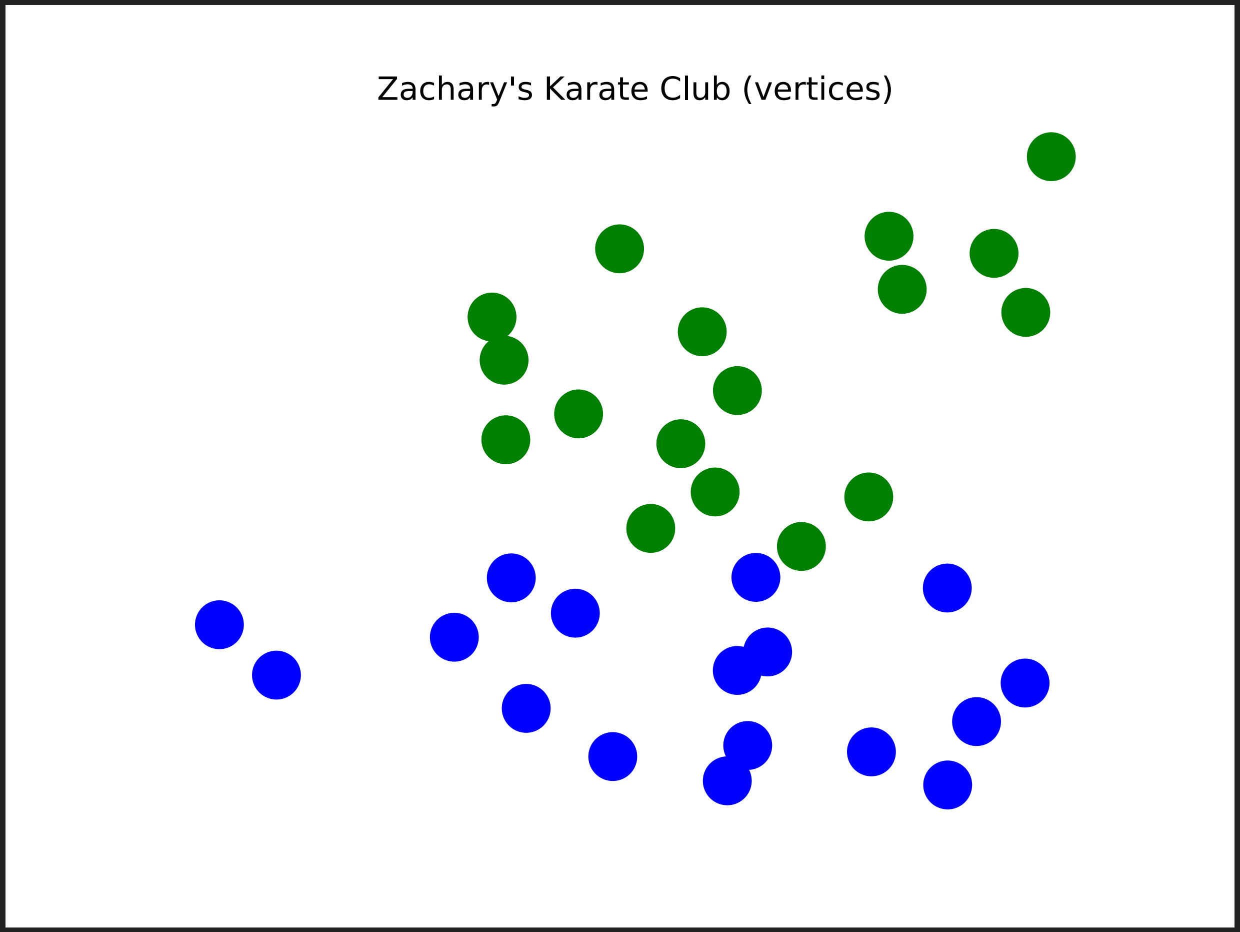
BASIC GRAPH NOTATION

Zachary's karate club graph is a social network of friendships among members of a karate club.

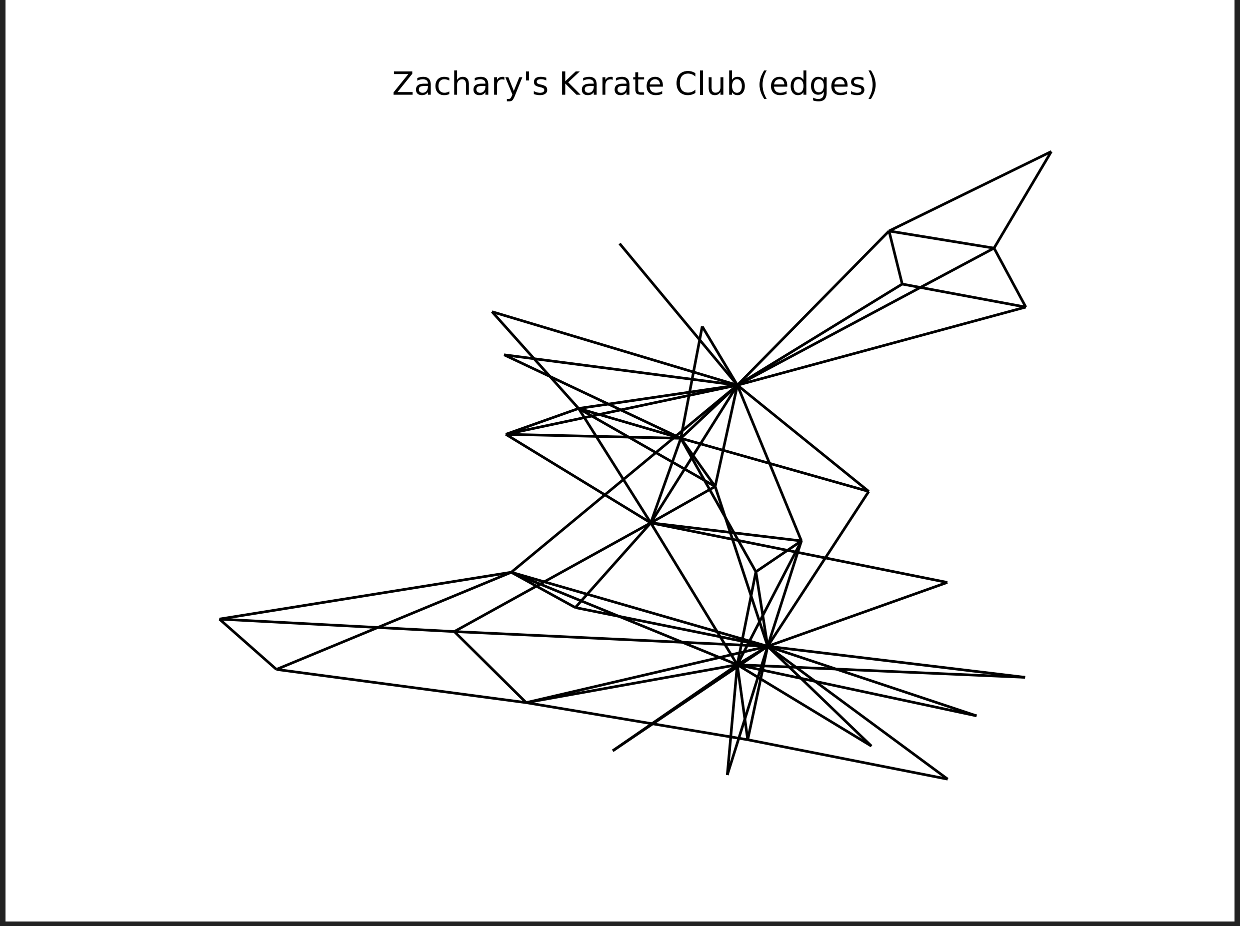
Due to a dispute, the original club disbanded and two new groups (colored green and blue) formed.



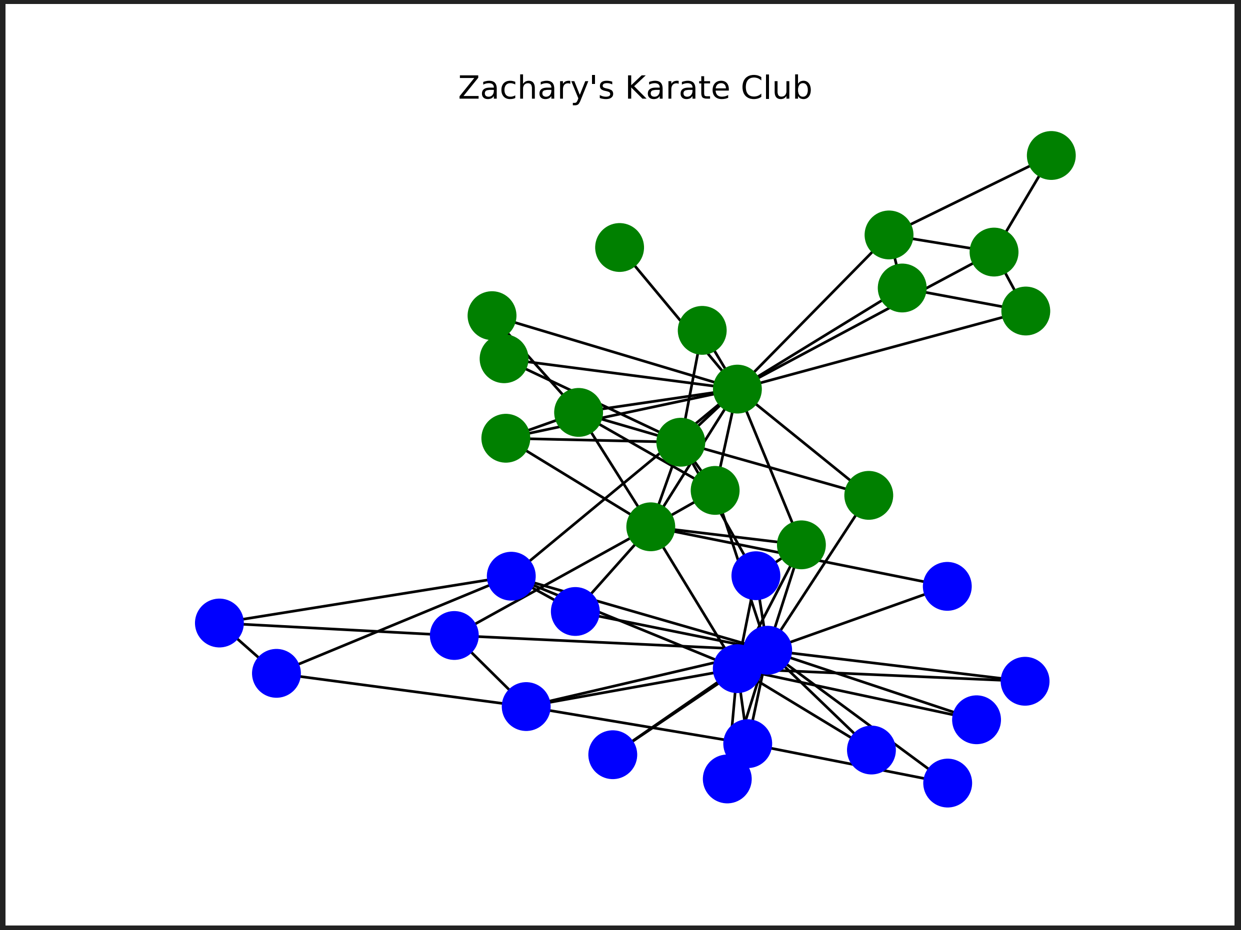
$\mathbb{V}$ = A set of vertices



$\mathbb{E} =$ A set of edges



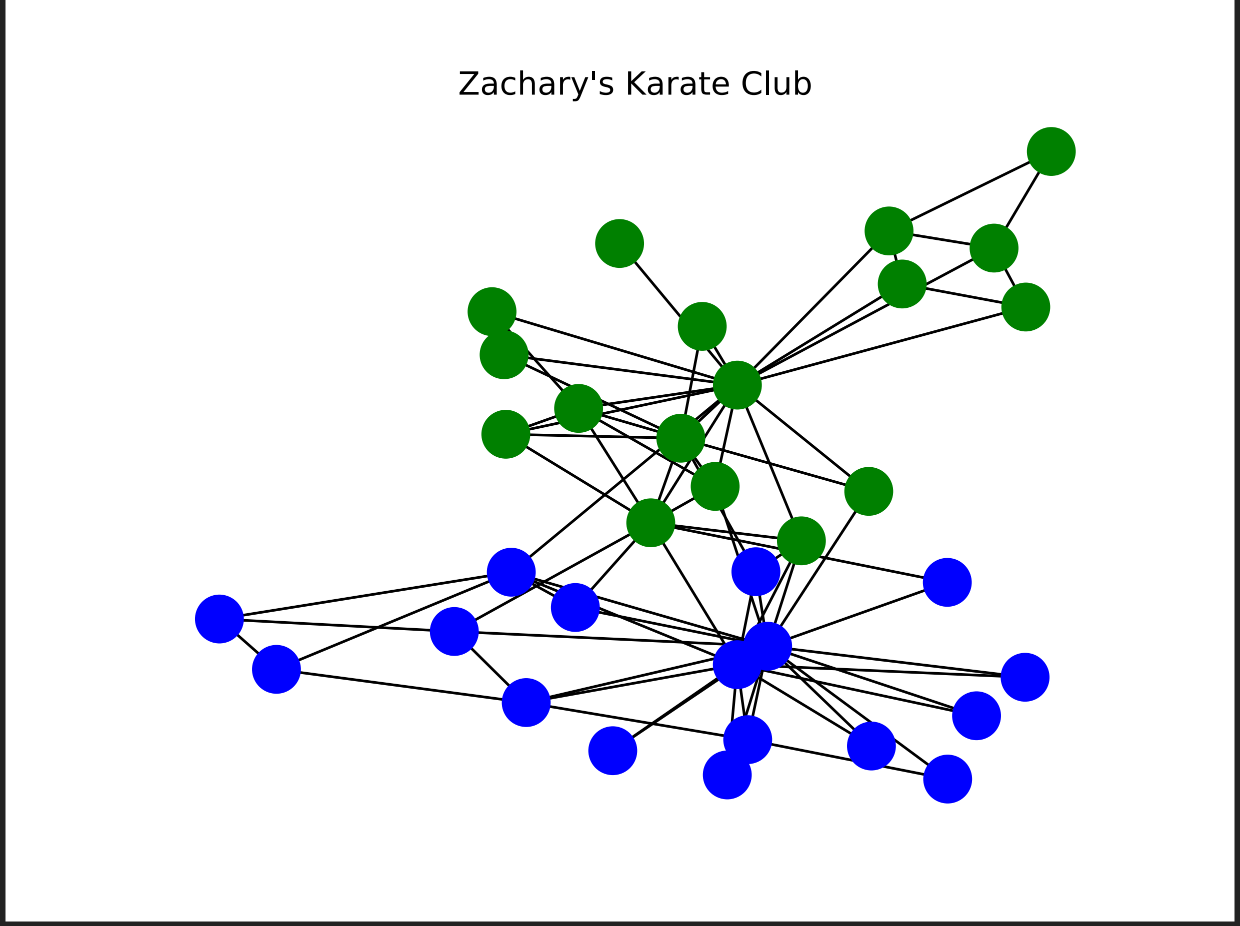
$$\mathbb{G} = (\mathbb{V}, \mathbb{E})$$



TASKS WITH GRAPHS

- Ranking (usually of vertices, can be of edges)
  - Find x_i in R for each vertex i
  - Degree centrality, eigenvector centrality, Katz centrality, PageRank
  - *Very* loosely: unsupervised regression
- Graph Partitioning
  - Partition vertices into two disjoint sets
  - Example: Spectral partitioning (Fiedler method)
- Graph Clustering or Community Detection
  - Separate vertices into multiple disjoint sets
  - Example: Spectral modularity maximization
  - *Very* loosely: unsupervised classification

WHAT MAKES A VERTEX IMPORTANT?



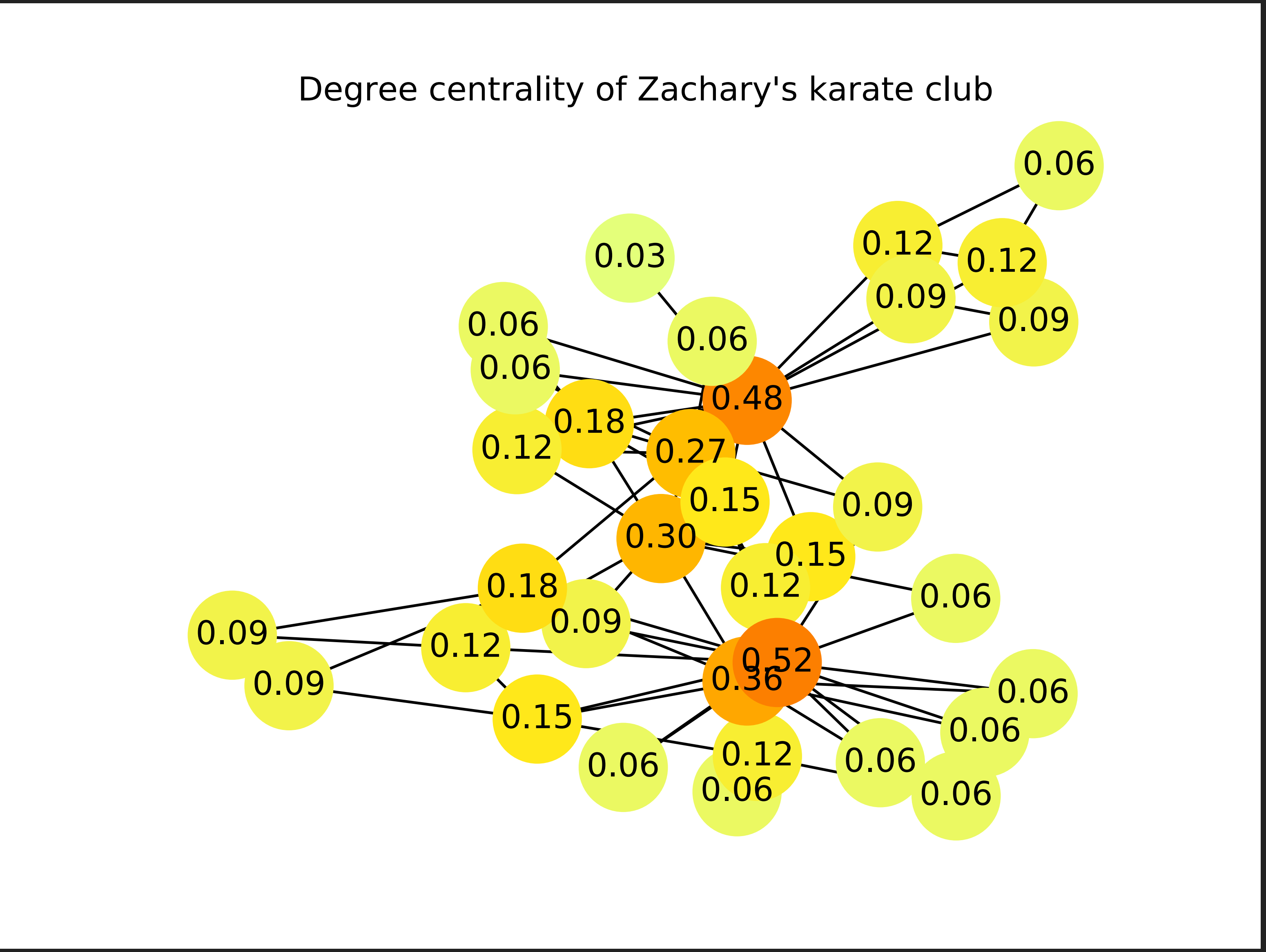
Based only on the structure of the social network before the bifurcation, can we predict with high accuracy which group someone would join?

DEGREE CENTRALITY

The more important a vertex is, the more vertices it is connected to.

Degree centrality measures the importance of a vertex according to this criterion.

For any graph with n vertices, the degree of a vertex is normalized by n-1, the maximum number of edges a vertex can have in any (simple) graph. More complex graphs, such as those with self loops, degree centrality can be > 1.

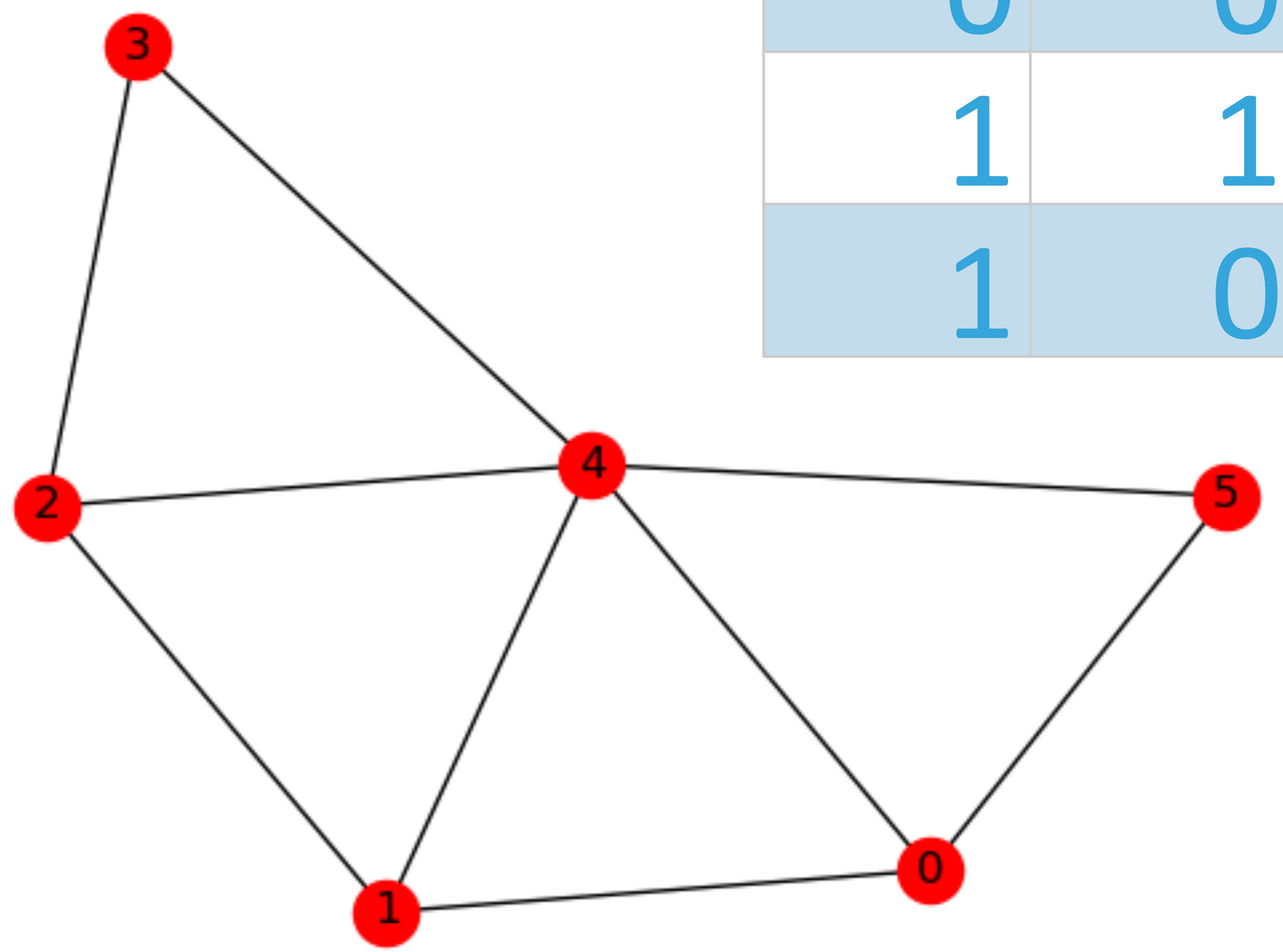


A vertex with low degree that is connected only to high-degree vertices has the same degree centrality as a vertex with low degree connected only to low-degree vertices. Is that correct? Why or why not?

ADJACENCY MATRIX

$$a_{ij} = \begin{cases} 1 \ \text{if} \ (i,j) \in E \\ 0 \ \text{otherwise} \end{cases}$$

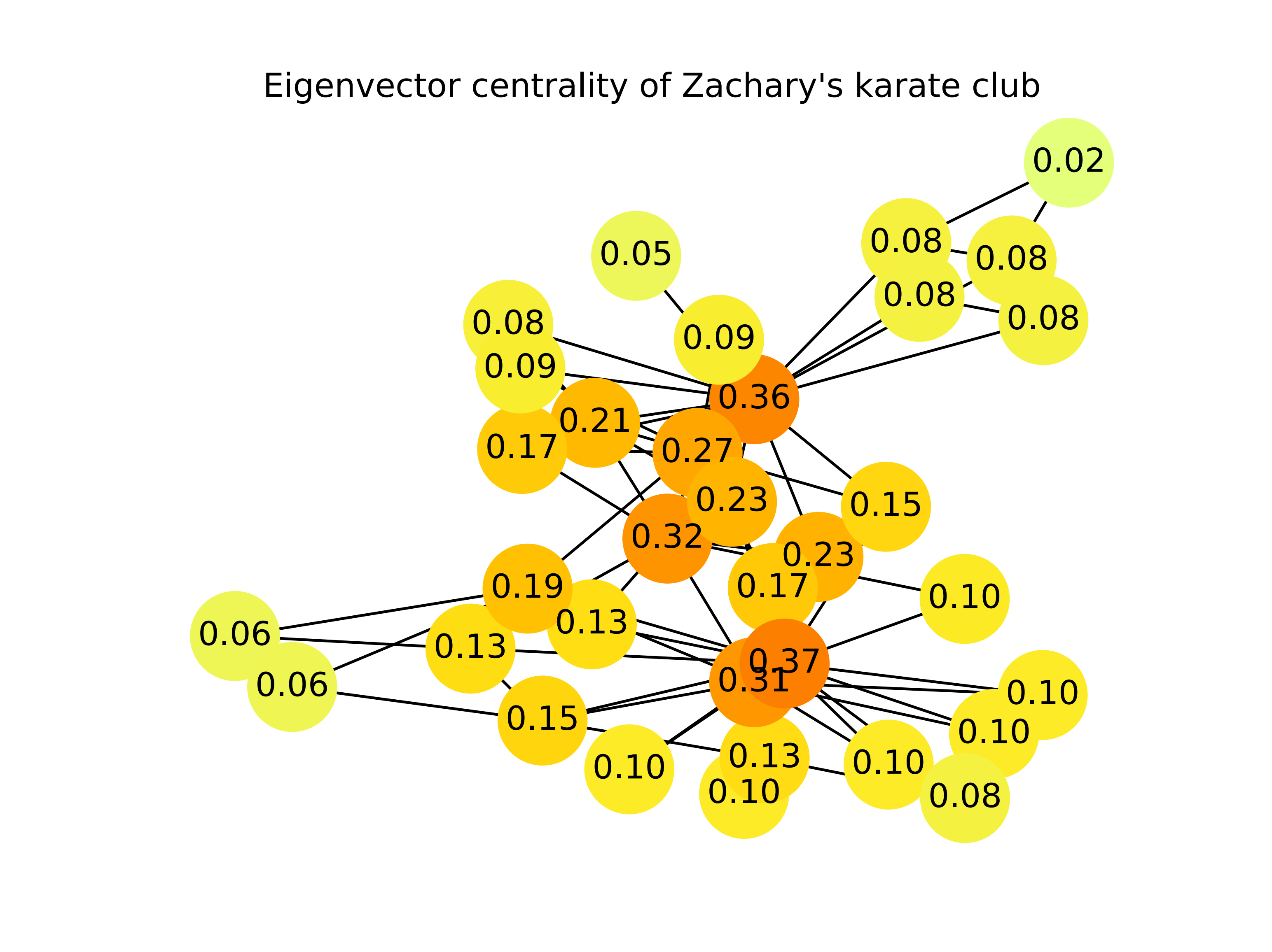
0	1	0	0	1	1
1	0	1	0	1	0
0	1	0	1	1	0
0	0	1	0	1	0
1	1	1	1	0	1
1	0	0	0	1	0



Newman-Watts-Strogatz graph with $k = 2$ and $p = 0.5$.

EIGENVECTOR CENTRALITY

Eigenvector centrality takes the *neighborhood* of a vertex into account, so low-degree vertices in a neighborhood consisting of high-degree vertices gets a high score.




```python
# coding: utf-8

import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
import scipy.linalg

G = nx.karate_club_graph()
title = "Zachary's karate club graph"
A = nx.adjacency_matrix(G).toarray()
w, v = scipy.linalg.eigh(A)
leading_eigenvector = np.abs(v[:, -1])
eigcent = dict(zip(
    range(len(w)),
    map('{:.2f}'.format, leading_eigenvector)
))
nx.draw_networkx(G,
    pos=nx.drawing.spring_layout(G, seed=0), labels=eigcent,
    node_size=1000, node_color=leading_eigenvector, cmap='Wistia')
plt.axis('off')
plt.title("Eigenvector centrality of Zachary's karate club")
plt.show(block=False)
```

The eigenvector centrality of vertex i is the i-th element of the leading eigenvector (the eigenvector corresponding to the largest, most positive eigenvalue).

EFFICIENTLY COMPUTING EIGENVECTOR CENTRALITY

The power method is an efficient, iterative algorithm for computing the leading eigenvector of a square matrix.

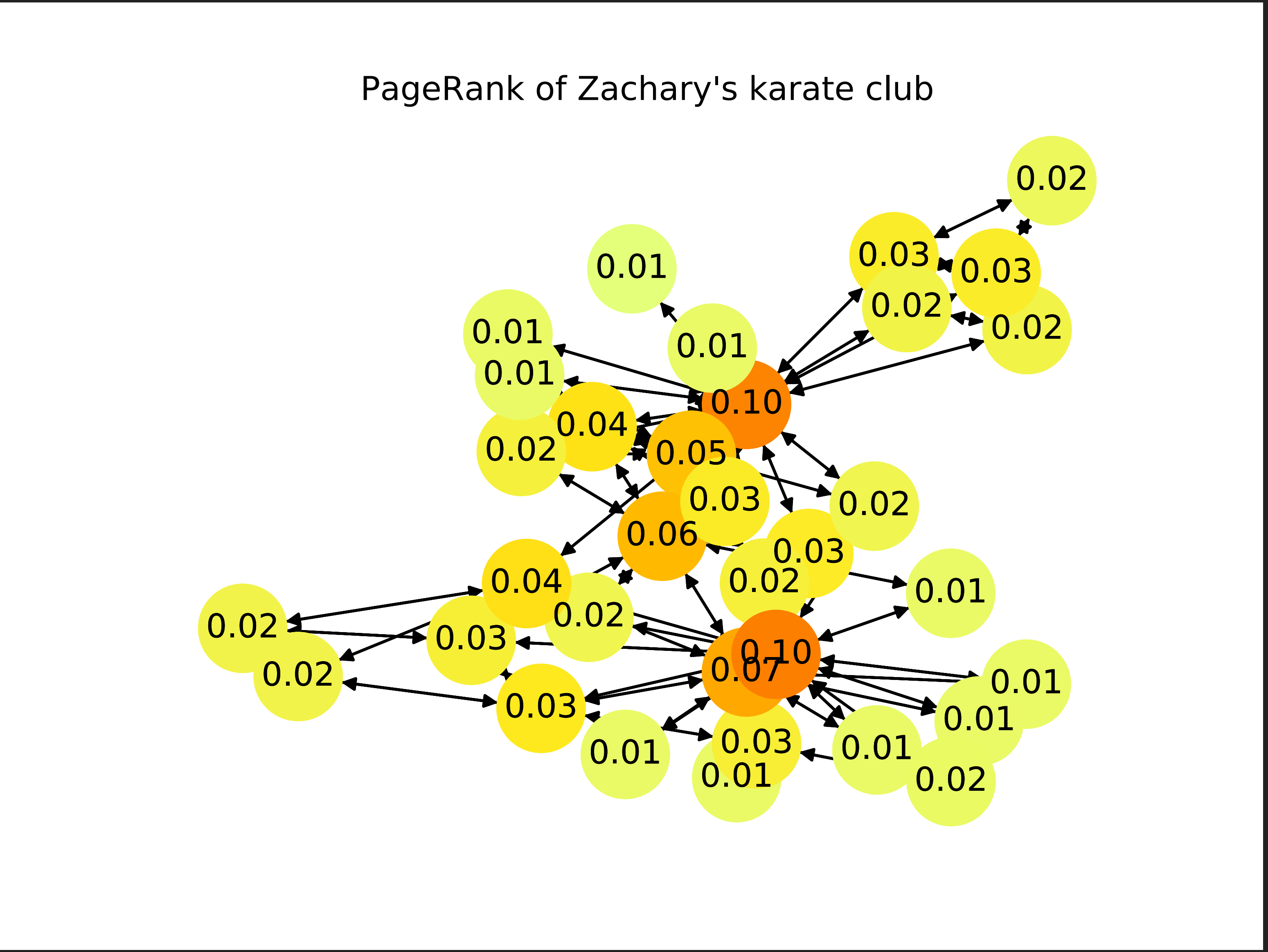
```
# coding: utf-8

import numpy as np
from sklearn.utils import check_random_state


def power_method(A, n_iters=100, random_state=None):
    """
    Compute eigenvector centrality via the power method.
    """
    random_state = check_random_state(random_state)
    ev = random_state.uniform(0.1, 1.0, size=(A.shape[0], 1))
    for i in range(n_iters):
        ev = A.T@ev
        ev = ev / np.linalg.norm(ev)
    return ev[:, 0]
```

The adjacency matrix of a large graph often cannot fit into memory on a single machine. A sparse representation is usually used.

PAGERANK

PageRank is, effectively, eigvenvector centrality for directed graphs with features to ensure the algorithm behaves well with e.g. vertices with no out-edges.

PageRank was crucial for Google early on. Eventually, it became a feature in a ranking model.




```
# coding: utf-8

import networkx as nx
import matplotlib.pyplot as plt

def pagerank(G, alpha=0.85):
    M = google_matrix(G, alpha)
    eigenvalues, eigenvectors = np.linalg.eig(M.T)
    ind = np.argmax(eigenvalues)
    # eigenvector of largest eigenvalue is at ind, normalized
    largest = np.array(eigenvectors[:, ind]).flatten().real
    norm = float(largest.sum())
    return dict(zip(G, map(float, largest / norm)))
```

WHAT'S THE MOST IMPORTANT FEATURE IN GOOGLE'S CURRENT RANKING MODEL?

WHAT'S THE MOST IMPORTANT FEATURE IN GOOGLE'S CURRENT RANKING MODEL?

Early in 2015, as Bloomberg recently reported, Google began rolling out a deep learning system called RankBrain that helps generate responses to search queries. As of October, RankBrain played a role in "a very large fraction" of the millions of queries that go through the search engine with each passing second.

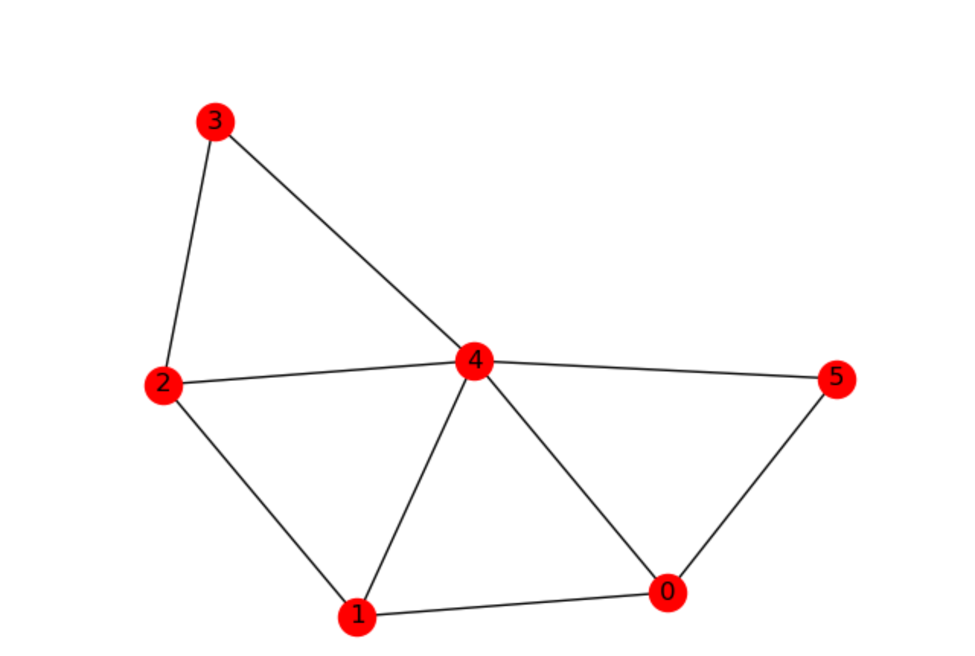
To be sure, deep learning is still just a part of how Google Search works. According to Bloomberg, RankBrain helps Google deal with about 15 percent of its daily queries—the queries the system hasn't seen in the past. Basically, this machine learning engine is adept at analyzing the words and phrases that make up a search query and deciding what other words and phrases carry much the same meaning. As a result, it's better than the old rules-based system when handling brand new queries—queries Google Search has never seen before.

AI Is Transforming Search. The Rest of the Web is Next.

Wired Business, 04 February 2016

POWERS OF THE ADJACENCY MATRIX

What aspect of a graph's structure is signified by the powers of its adjacency matrix?



Newman-Watts-Strogatz graph with $k = 2$ and $p = 0.5$.

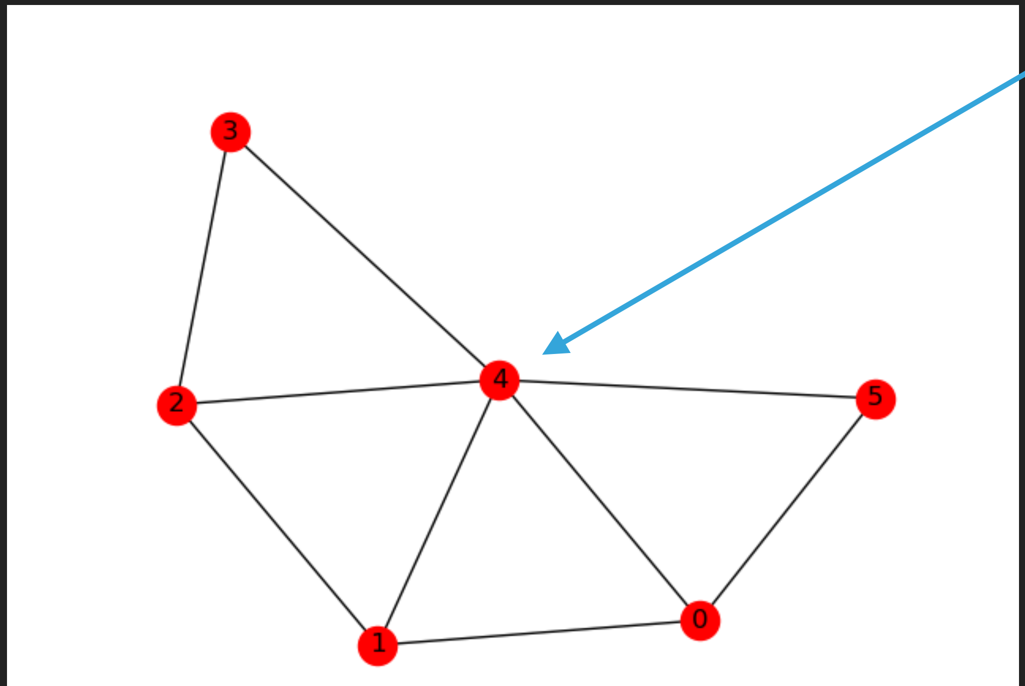
Here A^2 is A = AA, not element-wise exponentiation

A

0	1	0	0	1	1
1	0	1	0	1	0
0	1	0	1	1	0
0	0	1	0	1	0
1	1	1	1	0	1
1	0	0	0	1	0

A^2

3	1	2	1	2	1
1	3	1	2	2	2
2	1	3	1	2	1
1	2	1	2	1	1
2	2	2	1	5	1
1	2	1	1	1	2



Newman-Watts-Strogatz graph with $k = 2$ and $p = 0.5$.

Obviously, the diagonal contains the *degree* of each vertex.

$\mathbf{A}^2$

3	1	2	1	2	1
1	3	1	2	2	2
2	1	3	1	2	1
1	2	1	2	1	1
2	2	2	1	5	1
1	2	1	1	1	2

However, in an undirected graph with no self loops, the degree of a vertex is equivalent to the number of paths of length 2 from a given vertex back to itself.

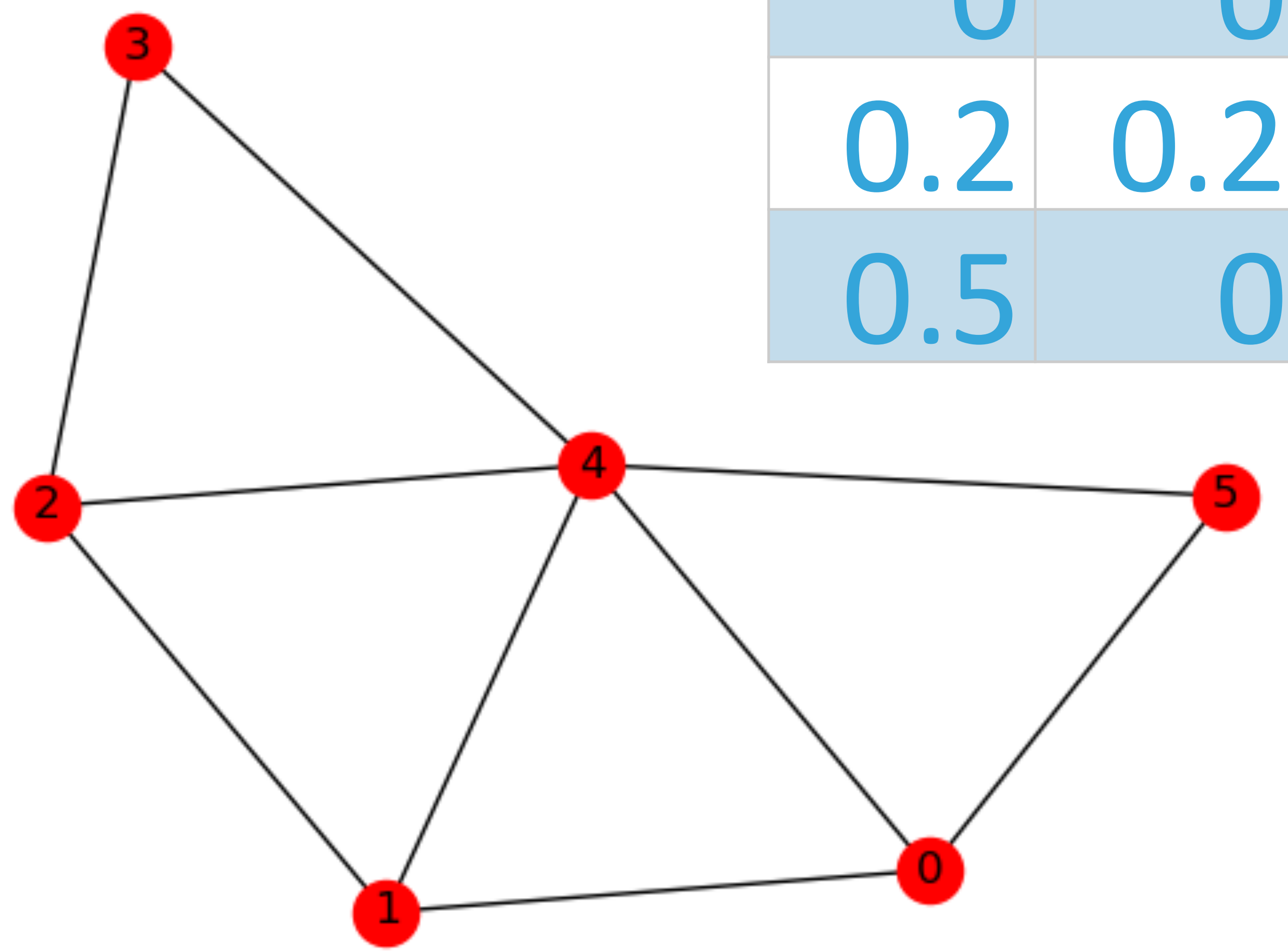
The diagonal of A^k contains the number of paths of length k from the vertex back to itself.

And the number of paths of length k from vertex i to vertex j is given by a_{ij}^k

THE TRANSITION PROBABILITY MATRIX

Dividing an adjacency matrix's rows by their sums yields a transition probability matrix P in which p_ij denotes the probability of transitioning from vertex i to vertex j on a random walk of the graph.

0	0.33	0	0	0.33	0.33
0.33	0	0.33	0	0.33	0
0	0.33	0	0.33	0.33	0
0	0	0.5	0	0.5	0
0.2	0.2	0.2	0.2	0	0.2
0.5	0	0	0	0.5	0



Newman-Watts-Strogatz graph with $k = 2$ and $p = 0.5$.

POWERS OF THE TRANSITION PROBABILITY MATRIX

With sufficiently large k, P^k reaches the *stationary distribution*, the result of which is that P^{k+1} = P^{k}P. (Sufficiency of k is a function of the diameter of the graph, the longest path.) What does p_{ij} denote?

Here P^2 is P = PP, not element-wise exponentiation

DIAG(P^K) ~ EIGENVECTOR CENTRALITY

$\mathbf{P}$

0	0.33	0	0	0.33	0.33
0.33	0	0.33	0	0.33	0
0	0.33	0	0.33	0.33	0
0	0	0.5	0	0.5	0
0.2	0.2	0.2	0.2	0	0.2
0.5	0	0	0	0.5	0

$\mathbf{P}^k$

0.17	0.17	0.17	0.11	0.28	0.11
0.17	0.17	0.17	0.11	0.28	0.11
0.17	0.17	0.17	0.11	0.28	0.11
0.17	0.17	0.17	0.11	0.28	0.11
0.17	0.17	0.17	0.11	0.28	0.11
0.17	0.17	0.17	0.11	0.28	0.11

POWERS OF THE TRANSITION PROBABILITY MATRIX

With sufficiently large k, P^k reaches the *stationary distribution*, the result of which is that P^{k+1} = P^{k}P. (Sufficiency of k is a function of the diameter of the graph, the longest path.) p_{ij} denotes the probability of ending up at vertex j on an infinite random walk from vertex i.

Here P^2 is P = PP, not element-wise exponentiation

DIAG(P^K) ~ EIGENVECTOR CENTRALITY

$$\mathbf{P}$$

0	0.33	0	0	0.33	0.33
0.33	0	0.33	0	0.33	0
0	0.33	0	0.33	0.33	0
0	0	0.5	0	0.5	0
0.2	0.2	0.2	0.2	0	0.2
0.5	0	0	0	0.5	0

$$\mathbf{P}^k$$

0.17	0.17	0.17	0.11	0.28	0.11
0.17	0.17	0.17	0.11	0.28	0.11
0.17	0.17	0.17	0.11	0.28	0.11
0.17	0.17	0.17	0.11	0.28	0.11
0.17	0.17	0.17	0.11	0.28	0.11
0.17	0.17	0.17	0.11	0.28	0.11