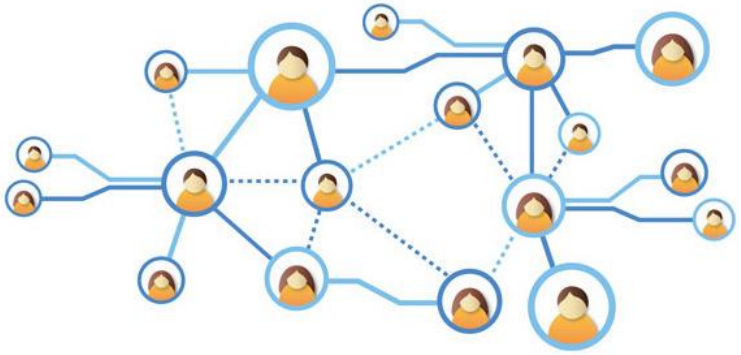
THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

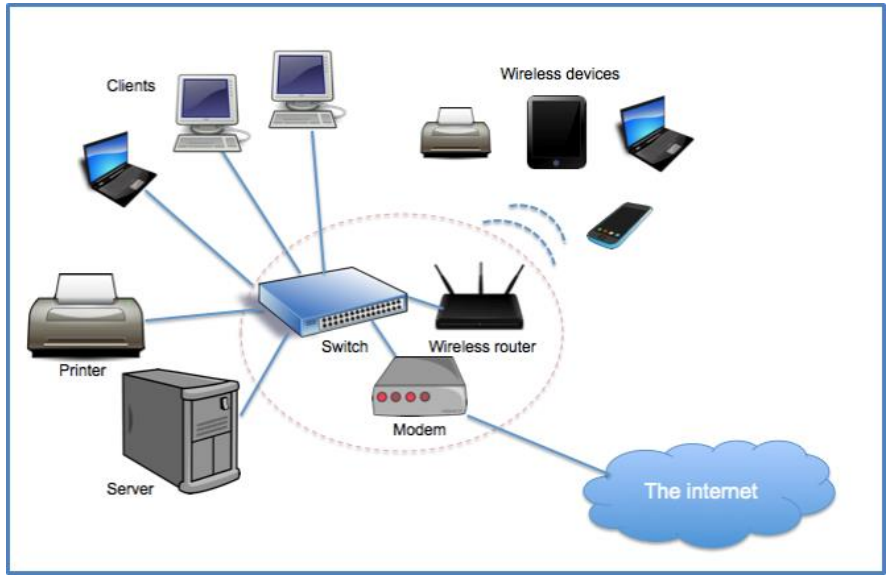
Graph Neural Networks

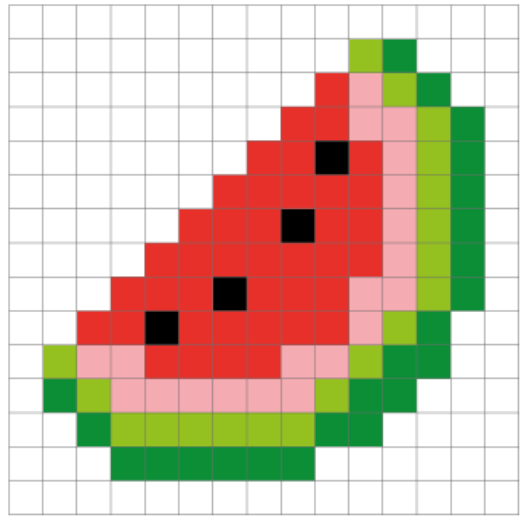
What they are, how to build them, and their applications

Isaiah J. King

(Almost) Everything is a Graph







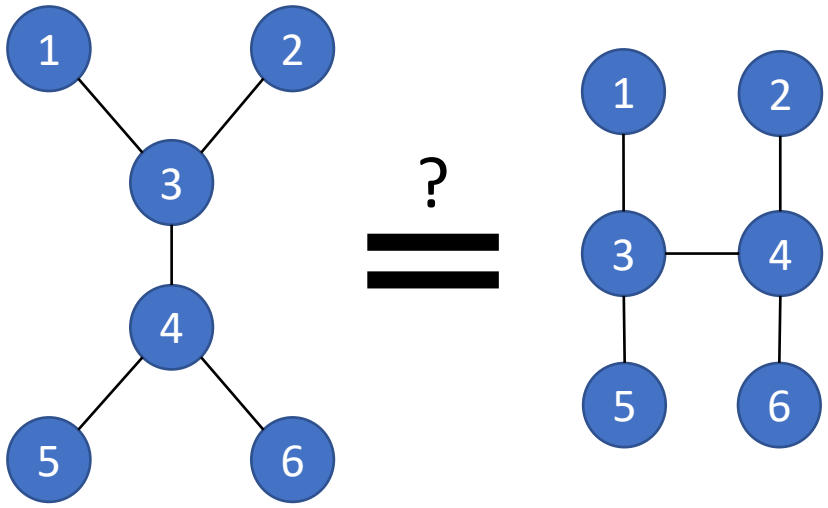
Social Networks

Computer Networks

Images

Img src (left to right)
https://blog.soton.ac.uk/skillted/2015/04/05/graph-theory-for-skillted/
https://www.schoolsofkingedwardvi.co.uk/ks2-computing-computing-theory-5-computer-networks/
https://projects.raspberrypi.org/en/projects/pixel-art-unplugged/1

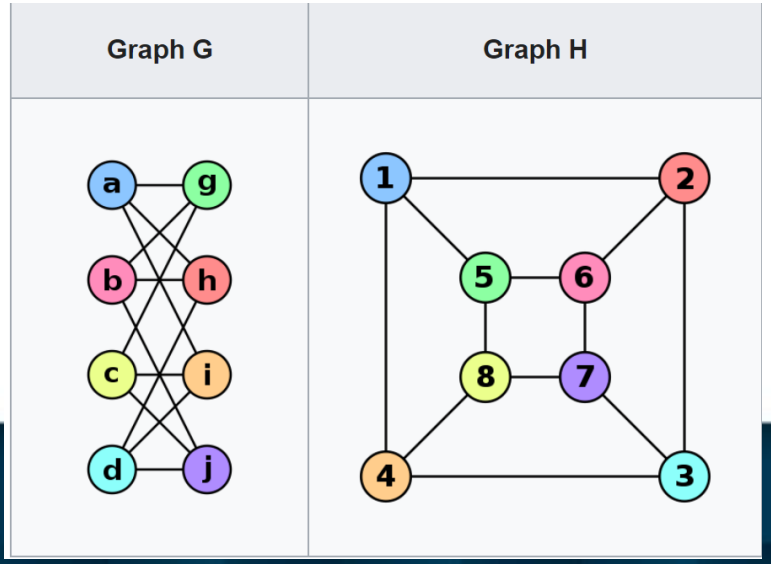
Graph Equality (aka isomorphisms)



Graph G=H iff

For all edges (u,v) in G

There exists (f(u), f(v)) in H

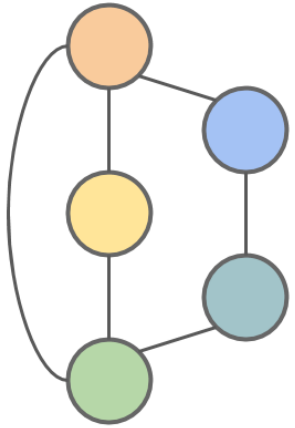


Very hard (NP-hard?) to find solutions

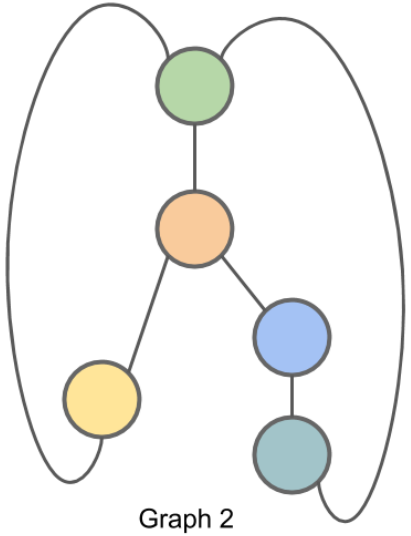
Img src: https://en.wikipedia.org/wiki/Graph_isomorphism

Graph Inequality

Weisfeiler-Lehman Isomorphism Test

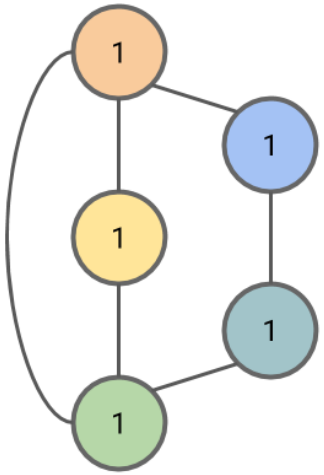


Graph 1

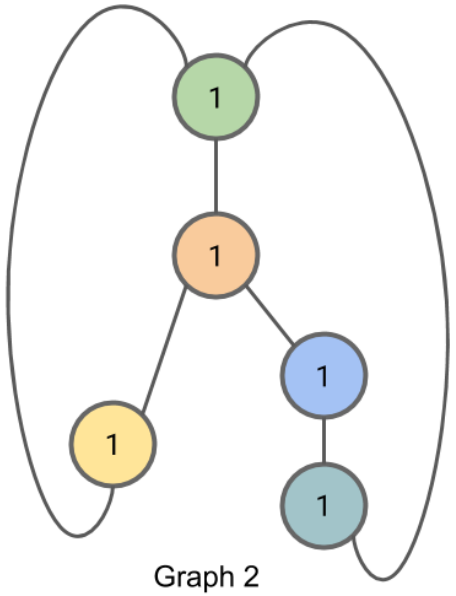


Graph 2

C_0



Graph 1

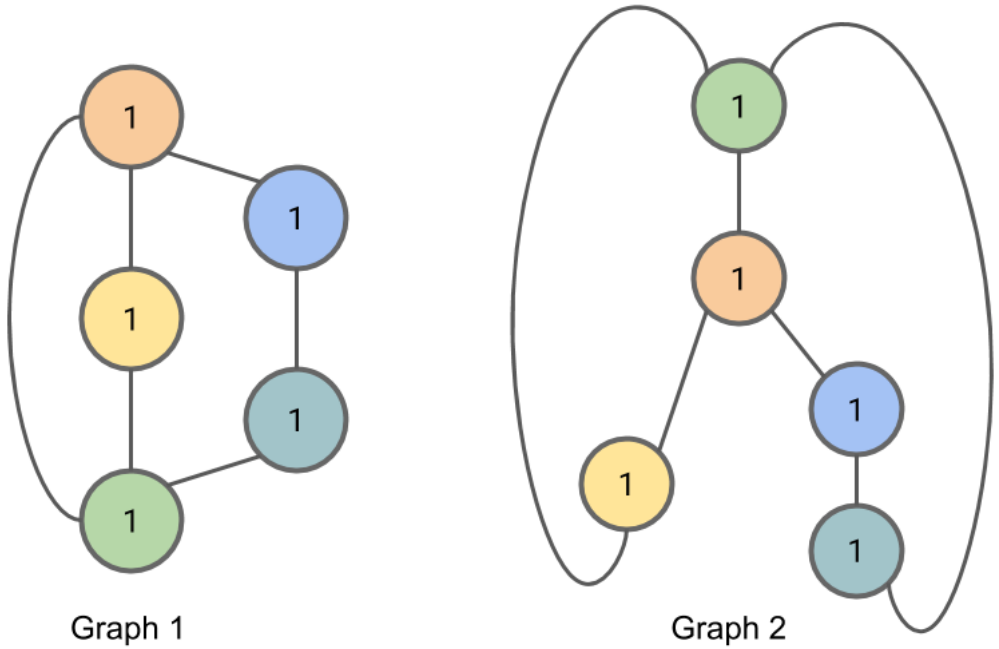


Graph 2

Images for all WL slides: https://davidbieber.com/post/2019-05-10-weisfeiler-lehman-isomorphism-test/

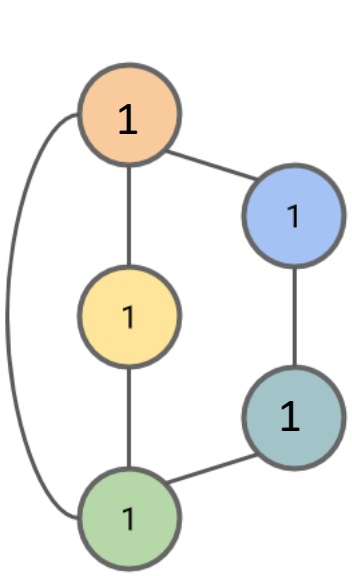
WL-Isomorphism Test (cont'd)

C_0

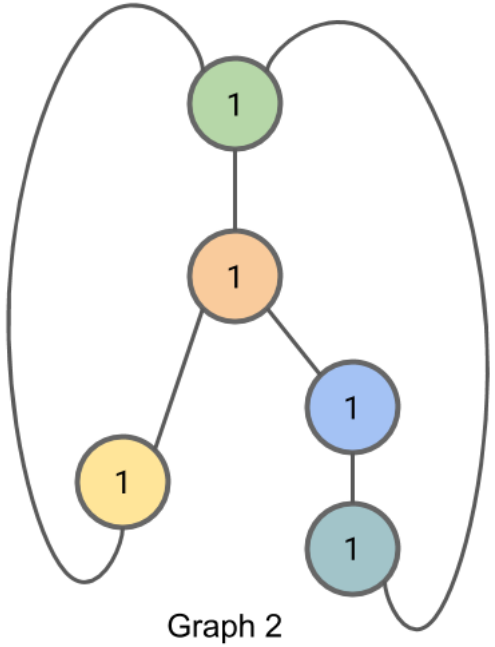


WL-Isomorphism Test (cont'd)

C_0

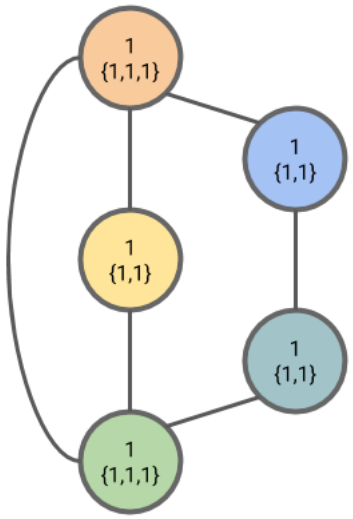


Graph 1

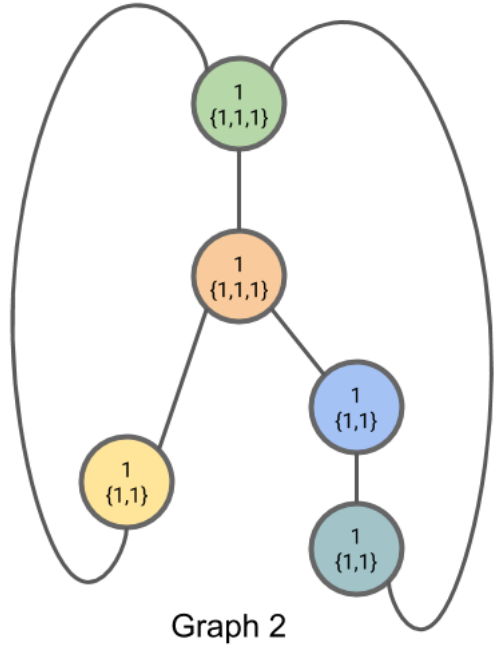


Graph 2

L_1



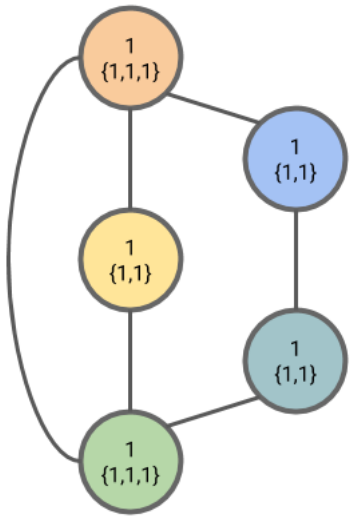
Graph 1



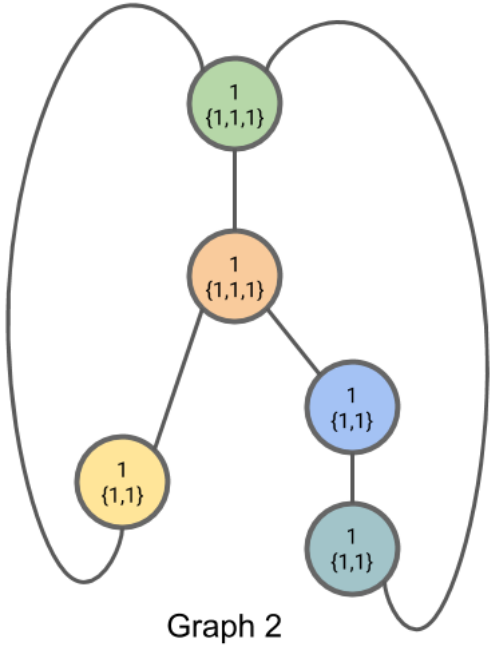
Graph 2

WL-Isomorphism Test (cont'd)

L_1

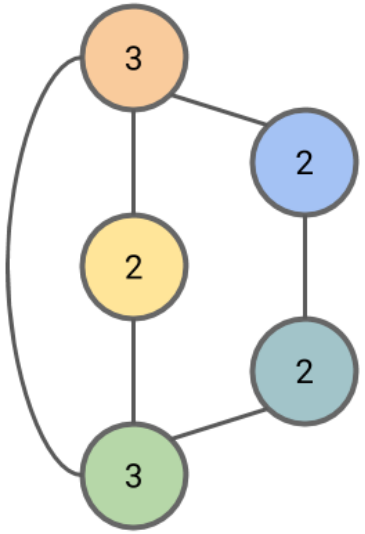


Graph 1

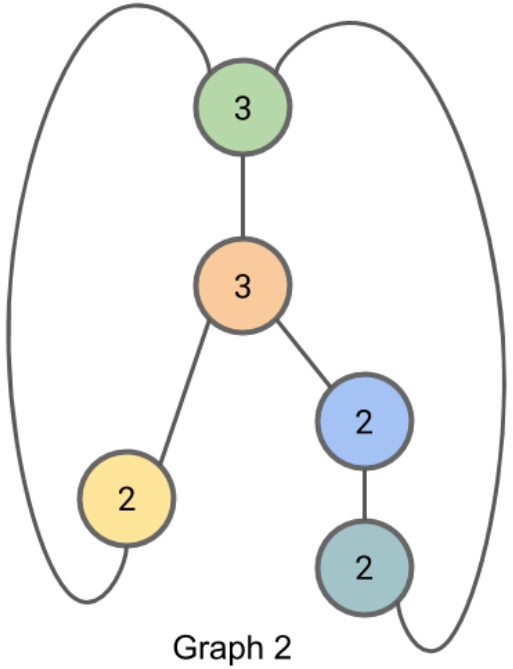


Graph 2

C_1

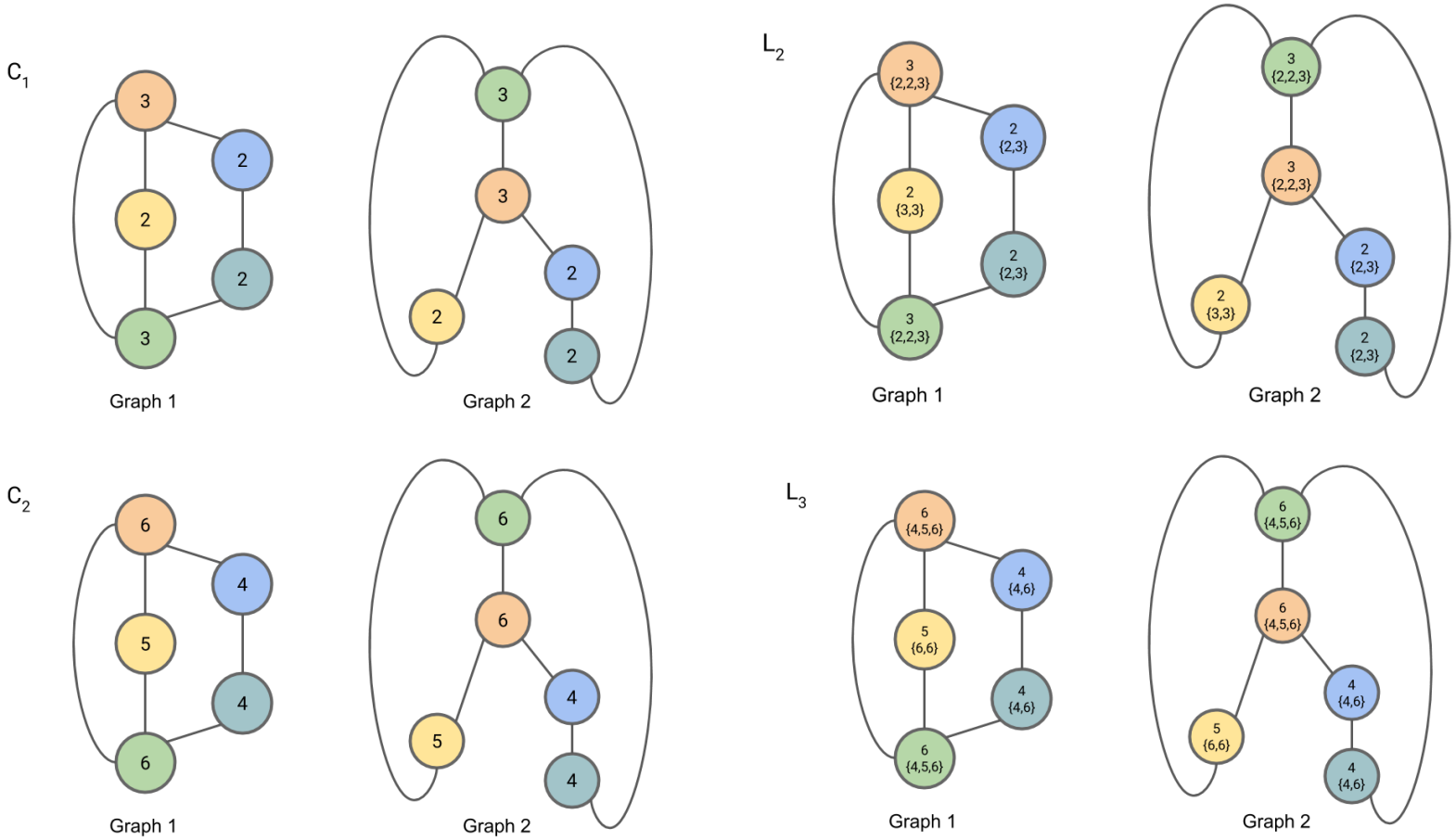


Graph 1

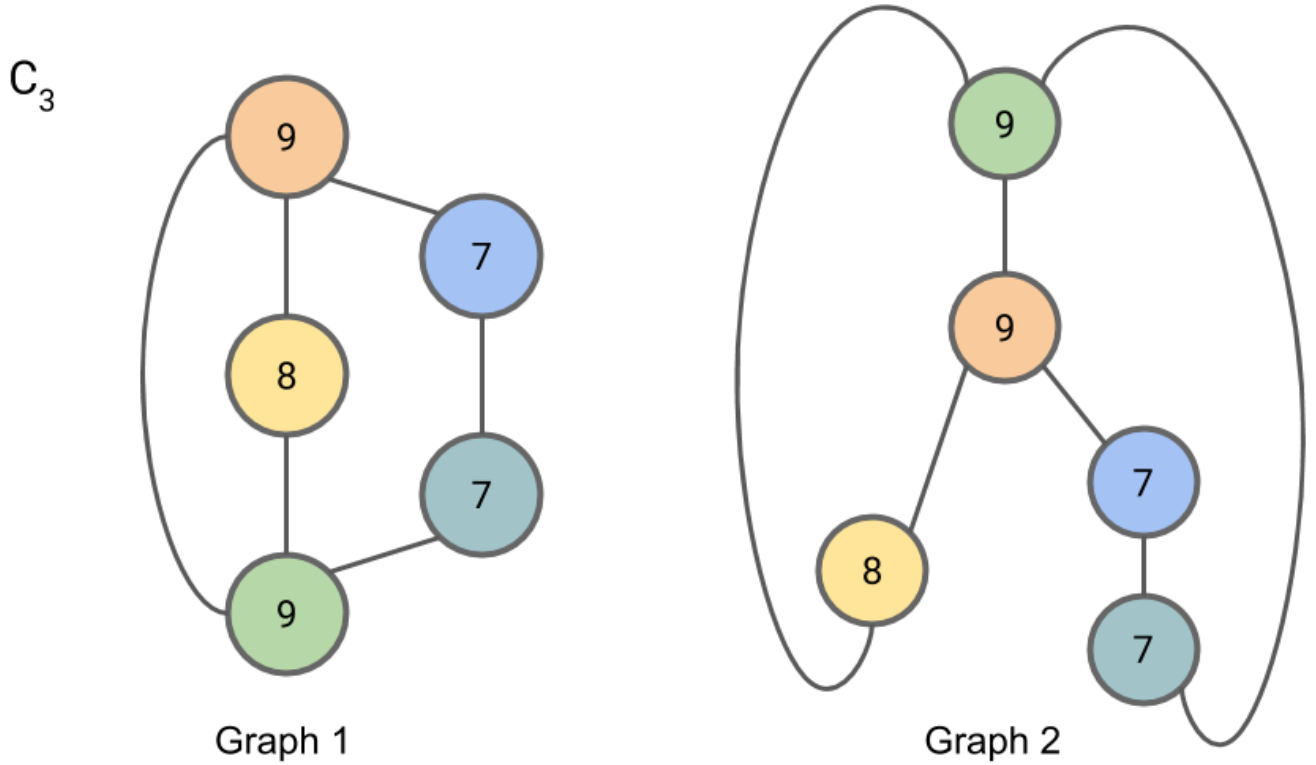


Graph 2

WL-Isomorphism Test (cont'd)

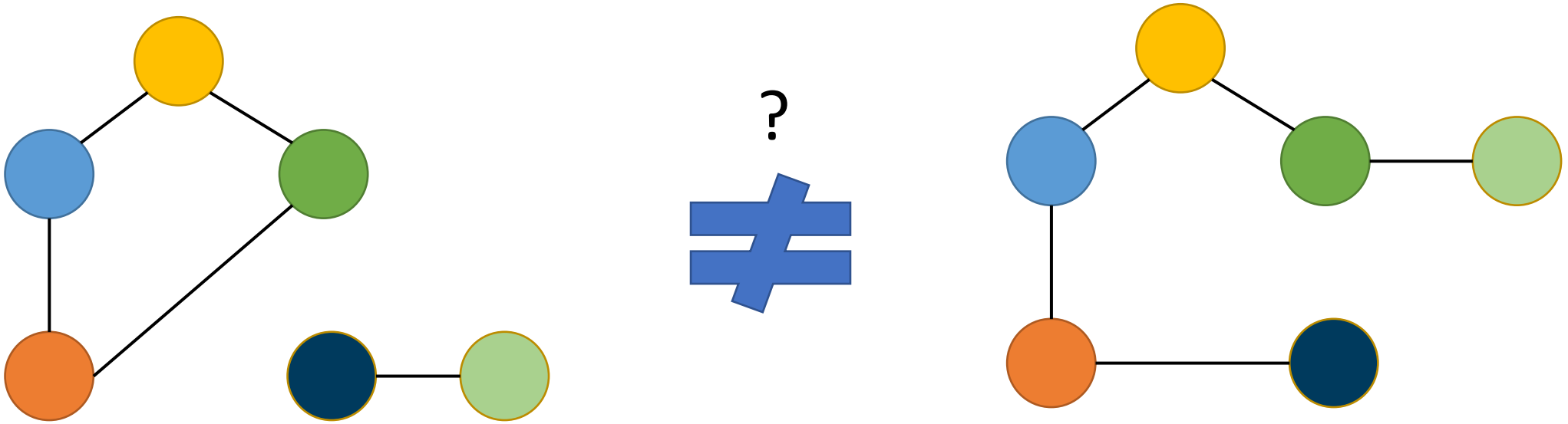


WL-Isomorphism Test (cont'd)

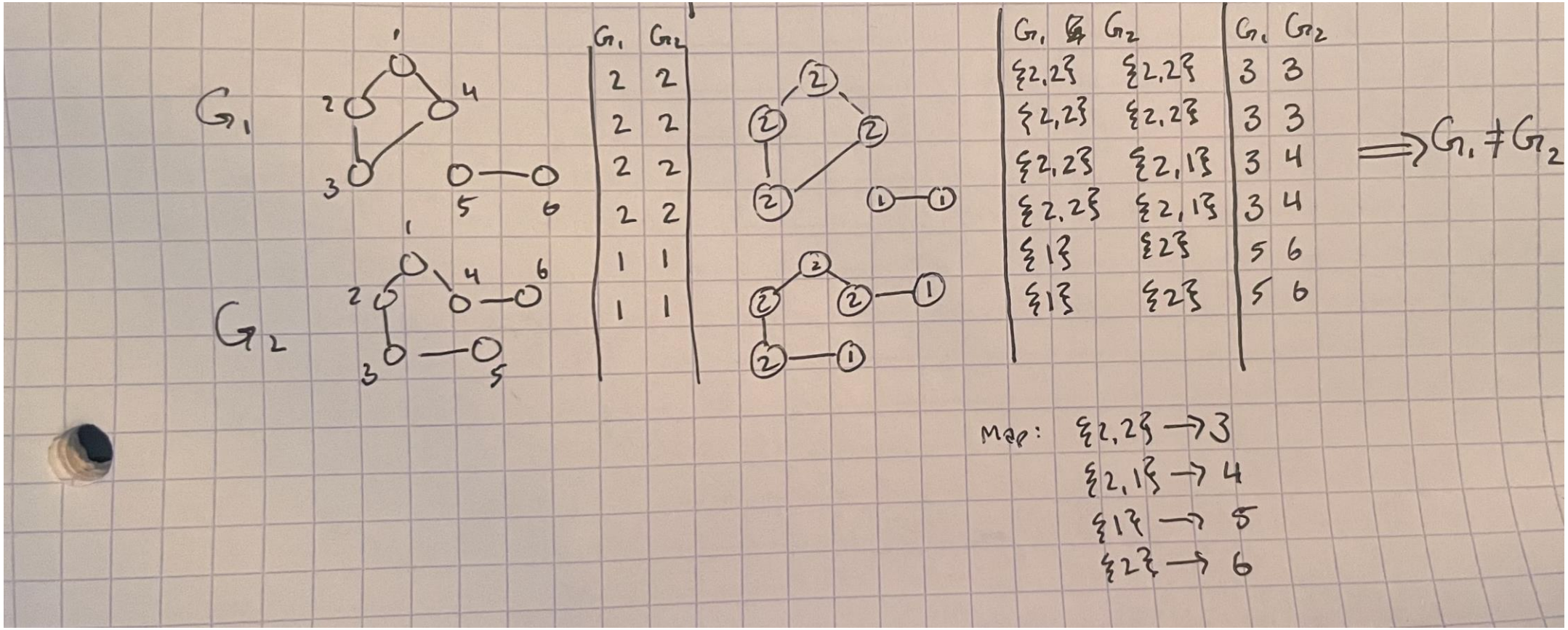


Graph	9	8	7
1	2	1	2
2	2	1	2

WL-Isomorphism Exercise



Solution



Wait, what does this have to do with GNNs?

- Graph summarization
  - Condensing the information of a graph into a single vector
  - Comparing graph vectors tells us something about the two graphs
- Node representation
  - Iteration 1: how many edges each node has
  - Iteration 2: how many edges each node's neighbor has
- Message passing
  - Important mechanism for GNNs

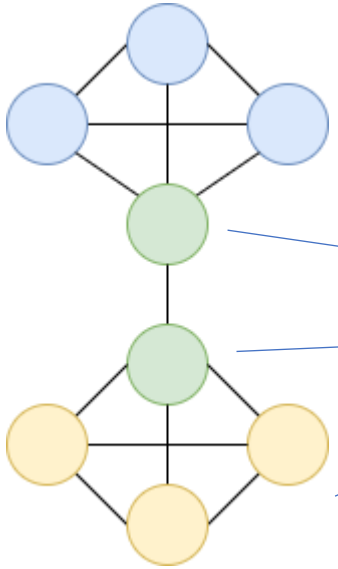
Message Passing Neural Networks

- General term for family of GNNs
- Extends WL test
  - Pass vectors instead of numbers (also called WL-n test)
  - Aggregate and nonlinear transform instead of hashing
  - Allows for nodes to have input features

$$h_v^{(\ell)} = f^{(\ell)}\left(\text{Aggregate}\left(h_u^{(\ell-1)} \mid u \in \mathcal{N}(v)\right)\right)$$

Example

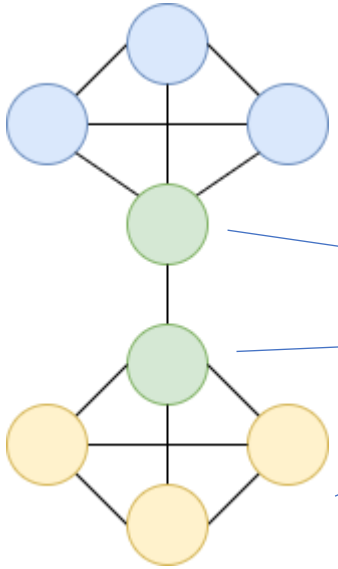
Aggregate(.) = Mean; f(x) = sigmoid(x)



Trek	LOTR
1.0	-0.5
0.5	0.5
0.5	0.5
-0.5	1.0

Example

Aggregate(.) = Mean; f(x) = sigmoid(x)



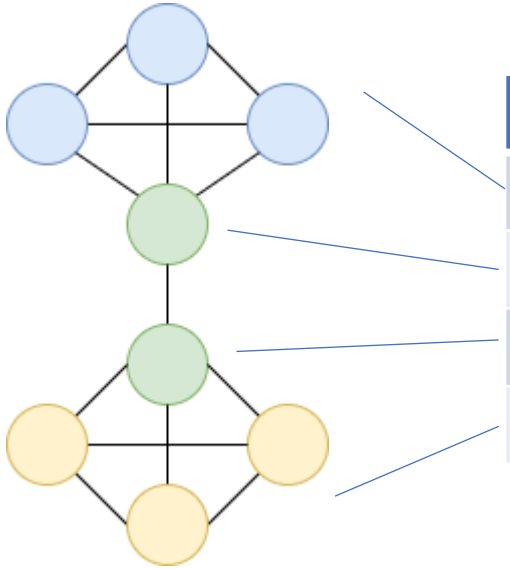
Trek	LOTR
1.0	-0.5
0.5	0.5
0.5	0.5
-0.5	1.0

Aggregate

Trek	LOTR
5/6	-1/6
7/8	-1/4
-1/4	7/8
-1/6	5/6

Example

Aggregate(.) = Mean; f(x) = sigmoid(x)



Trek	LOTR
1.0	-0.5
1.0	0.5
0.5	1.0
-0.5	1.0

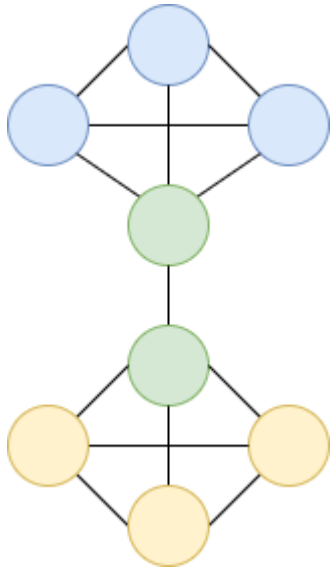
Aggregate

Trek	LOTR
5/6	-1/6
7/8	-1/4
-1/4	7/8
-1/6	5/6

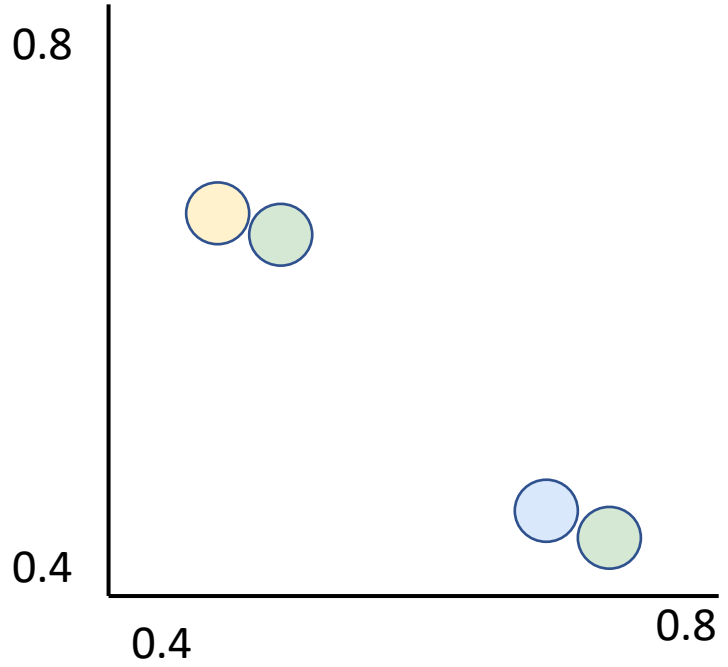
sigmoid

Trek	LOTR
0.70	0.46
0.71	0.43
0.43	0.71
0.46	0.70

Example



Trek	LOTR
0.70	0.46
0.71	0.43
0.43	0.71
0.46	0.70



Some popular GNNs

Graph Convolutional Networks (GCN)

- One of the earliest MPNNs, still the go-to

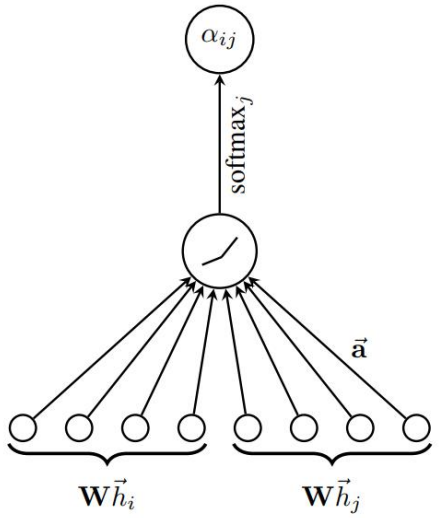
$$\mathbf{H}^{(\ell)} = \sigma\left(\mathbf{D}^{\frac{1}{2}}\mathbf{A}\mathbf{D}^{\frac{1}{2}}\mathbf{H}^{(\ell-1)}\mathbf{W}^{(\ell)}\right)$$

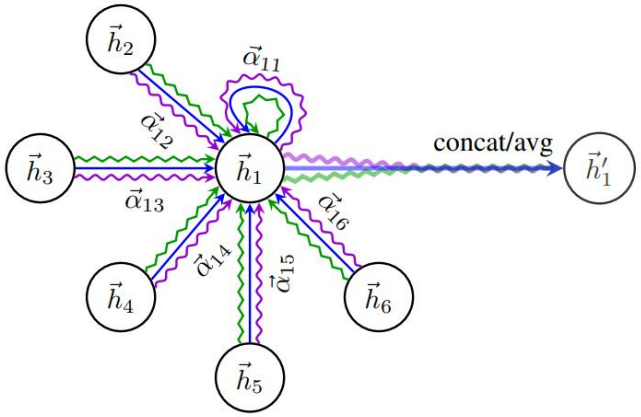
- D^½ A D^½ is the norm. adjacency matrix (calculated once)
- By multiplying features by adj. mat., we're efficiently doing mean message passing
- Finally, multiply aggregated result by trainable param W and sigmoid result

Very similar to image Conv. Mathematically same as the convolution operator if node feats treated as signals

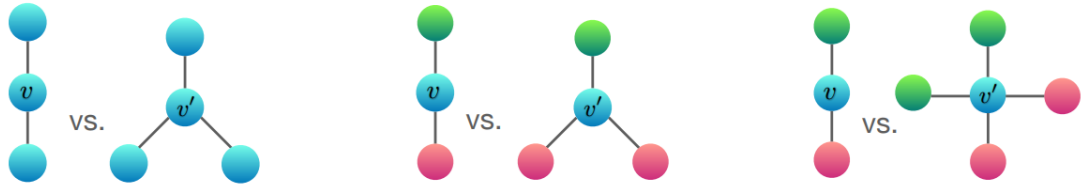
Graph Attention Networks (GAT)

- During the aggregation phase multiply node signals by trainable param *a*
- Essentially does a weighted average instead of mean
- Train more than one *a* parameter to get “multihead attention”





Graph Isomorphism Networks (GIN)



(a) Mean and Max both fail (b) Max fails (c) Mean and Max both fail

Figure 3: **Examples of graph structures that mean and max aggregators fail to distinguish.** Between the two graphs, nodes v and v' get the same embedding even though their corresponding graph structures differ. Figure 2 gives reasoning about how different aggregators "compress" different multisets and thus fail to distinguish them.

$$h_v^{(k)} = \text{MLP}^{(k)}\left(\left(1+\epsilon^{(k)}\right)\cdot h_v^{(k-1)} + \sum_{u\in\mathcal{N}(v)} h_u^{(k-1)}\right). \tag{4.1}$$

Generally, there may exist many other powerful GNNs. GIN is one such example among many maximally powerful GNNs, while being simple.

- Show that MPNNs are weaker than WL test
- Prove that using MLP (standard neural net) as aggregator creates maximally expressive GNN

Uses of GNNs

- Graph classification
  - Medical: Drug discovery, protein folding (molecules as graphs)
  - Images: Pixels as nodes with neighbors as adjacent pixels
  - NLP: nodes are words, edges are adjacent ones
- Node classification
  - Images: Finding pixels that make up certain objects
  - Databases: genre of movie given shared actors/paper given shared authors, etc.
- Link prediction
  - Cyber security: which computers shouldn't be talking to each other?
  - Knowledge Graphs: inference, which edges should exist?

Case study 1: Link prediction for cyber security

EULER: Detecting Network Lateral Movement via Scalable Temporal Link Prediction

Isaiah J. King and H. Howie Huang
George Washington University
{iking5, howie}@gwu.edu

***Abstract*—Lateral movement is a key stage of system compromise used by advanced persistent threats. Detecting it is no simple task. When network host logs are abstracted into discrete temporal graphs, the problem can be reframed as anomalous edge detection in an evolving network. Research in modern deep graph learning techniques has produced many creative and complicated models for this task. However, as is the case in many machine learning fields, the generality of models is of paramount importance for accuracy and scalability during training and inference. In this paper, we propose a formalized approach to this problem with a framework we call EULER. It consists of a model-agnostic graph neural network stacked upon a model-agnostic sequence encoding layer such as a recurrent neural network. Models built according to the EULER framework can easily distribute their graph convolutional layers across multiple machines for large performance improvements. Additionally, we demonstrate that EULER-based models are competitive, or better than many state-of-the-art approaches to anomalous link detection and prediction. As anomaly-based intrusion detection systems, EULER models can efficiently identify anomalous connections between entities with high precision and outperform other unsupervised techniques for anomalous lateral movement detection.**

The most robust way to detect malware propagation is not to exhaustively list every known malicious signature correlating with it; rather it is to train a model to learn what normal activity looks like, and to alert when it detects behavior that deviates from it. However, detecting anomalous activity in an enterprise network presents unique challenges. The data involved both during training and the implementation of an anomaly-based intrusion detection system is enormous. Often the log files that such a system would require as input are terabytes large. To be useful, a lateral movement detection model must be highly scalable to accommodate such large data. Additionally, when viewed as a classification problem, any such system would have to be highly precise. Millions of events occur in an enterprise network on any given day, and only a fraction of a percent of all interactions are ever anomalous [18]. Therefore, a model must have an extremely low rate of false alerts so as not to overwhelm its users.

In this work, we formulate anomalous lateral movement detection as a temporal graph link prediction problem. Interactions occurring in discrete units of time on a network can be abstracted into a series of graphs $G_t = \{V, S_t\}$ called

Network activity as a temporal graph

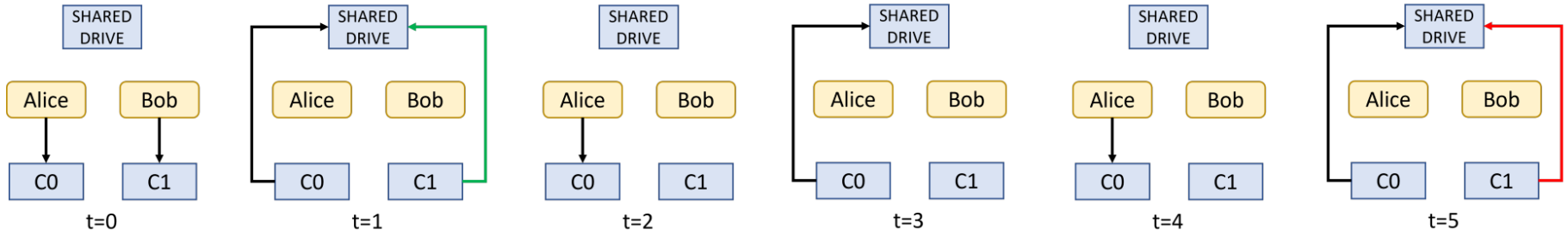
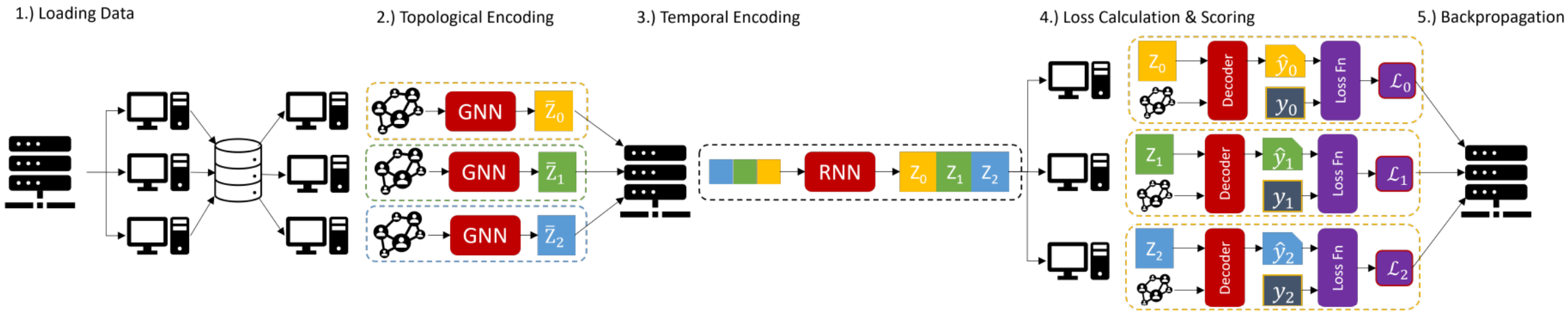


Fig. 2: A simple example of hard to detect temporally anomalous activity in a network with 3 machines and two users. The normal sequence of events is that a user authenticates with a computer, then that computer makes a request to the shared drive. However, at time t5, computer 1 makes a request to the shared drive without Bob first authenticating with it, denoting a perhaps malicious process running on computer 1 acting not on behalf of the machine's primary user.

Solution: distributed temporal link predictor



Case study 2: Node embedding for document recommendation

2022 IEEE International Conference on Big Data (Big Data)

GRAGGLE: A Graph-based Approach to Document Clustering

Isaiah J. King and H. Howie Huang
GraphLab
The George Washington University
Washington DC, USA
{iking5,howie}@gwu.edu

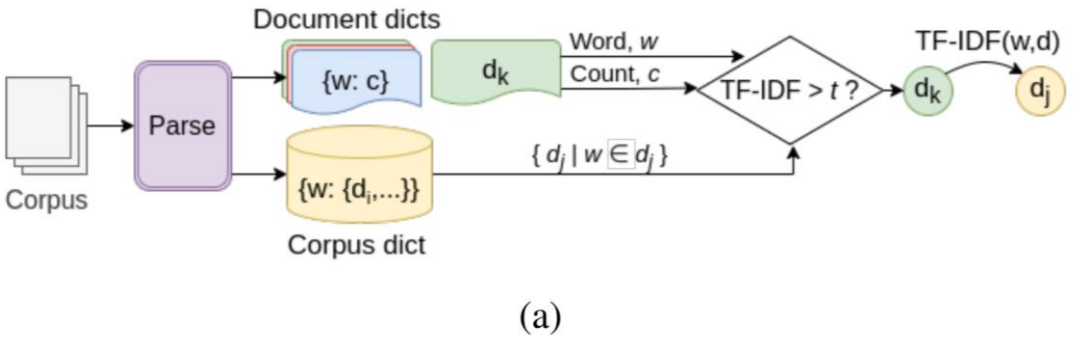
***Abstract*—Document recommendation systems have traditionally relied upon high-dimensional vector representations that scale poorly in corpora with diverse vocabularies. Existing graph-based approaches focus on the metadata of documents and, unfortunately, ignore the content of the papers. In this work, we have designed and implemented a new system we call GRAGGLE, which builds a graph to model a corpus. Nodes are papers, and edges represent significant words shared between them. We then leverage modern graph learning techniques to turn this graph into a highly efficient tool for dimensionality reduction. Documents are represented as low-dimensional vector embeddings generated with a graph autoencoder. Our experiments show that this approach outperforms traditional document vector-based and text autoencoding approaches on labeled data. Additionally, we have applied this technique to a repository of unlabeled research documents about the novel coronavirus to demonstrate its effectiveness as a real-world tool.**

***Index Terms*—Data mining, graph analytics, recommender systems, text mining**

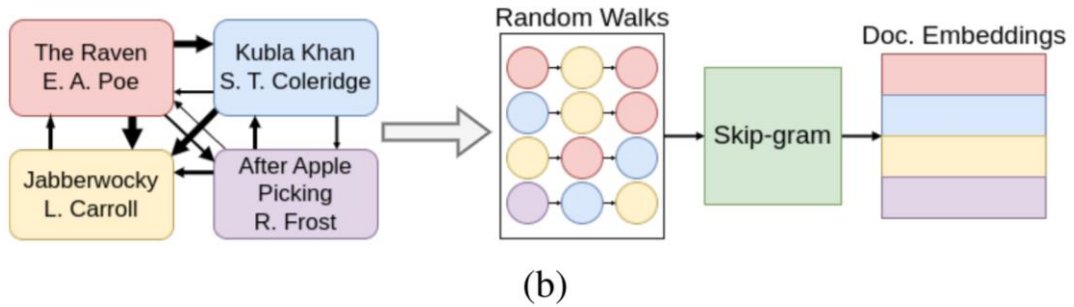
representing corpora as graphs, this view of documents as a collective rather than discrete units is realized. However, existing graph-based techniques are often based on relationships in the metadata of the papers such as citation networks. They are more often used for supervised learning [5]–[7]. Often these techniques use computationally expensive means of graph clustering such as geodesic distance [8]. Few exist which have graph structures derived from document text. Those that do, do not use a graph autoencoding technique [9]–[11] and lack the generality and expressiveness it offers.

To address these perceived shortcomings in the literature, we propose GRAGGLE[1], a novel approach to feature extraction from text that captures complex contextual relationships between documents. Rather than focusing on metadata, or using vectors generated just from the text, GRAGGLE generates

Solution: build graph of documents that share words



(a)



(b)

Applied to unlabled corpus of COVID-19 research papers

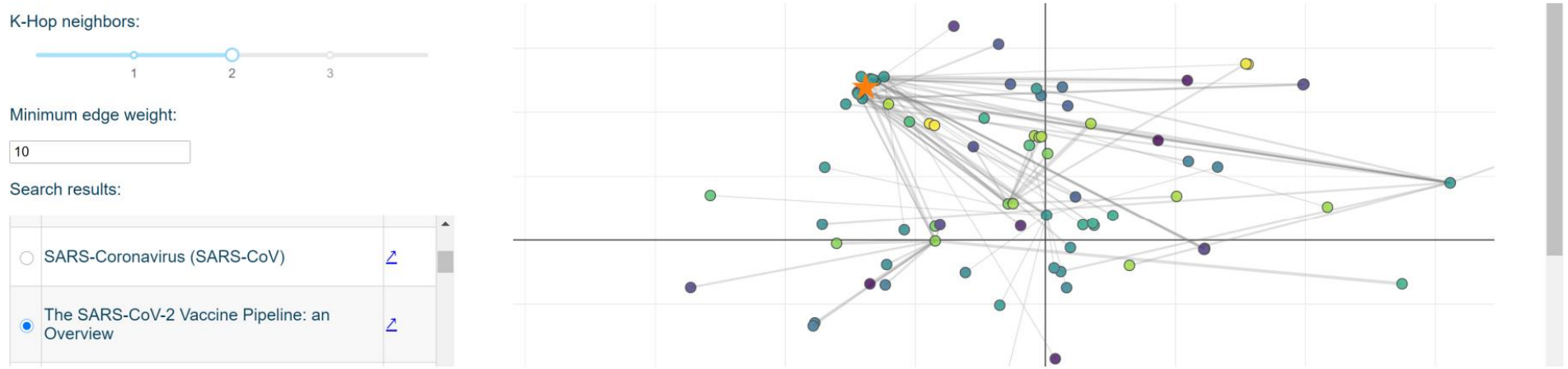


Fig. 4: The GRAGGLE search engine displaying t-SNE projected embeddings of the 2-hop neighbors of a search result. Different colored nodes represent different cluster assignments.

Your turn to try!

https://github.com/zazyzaya/CloudGNNIntro