Graph Analysis

Outline

Graph Analysis

Motivation:

- **Why analyze graphs?**

Graph basics

Graph analysis Algorithms

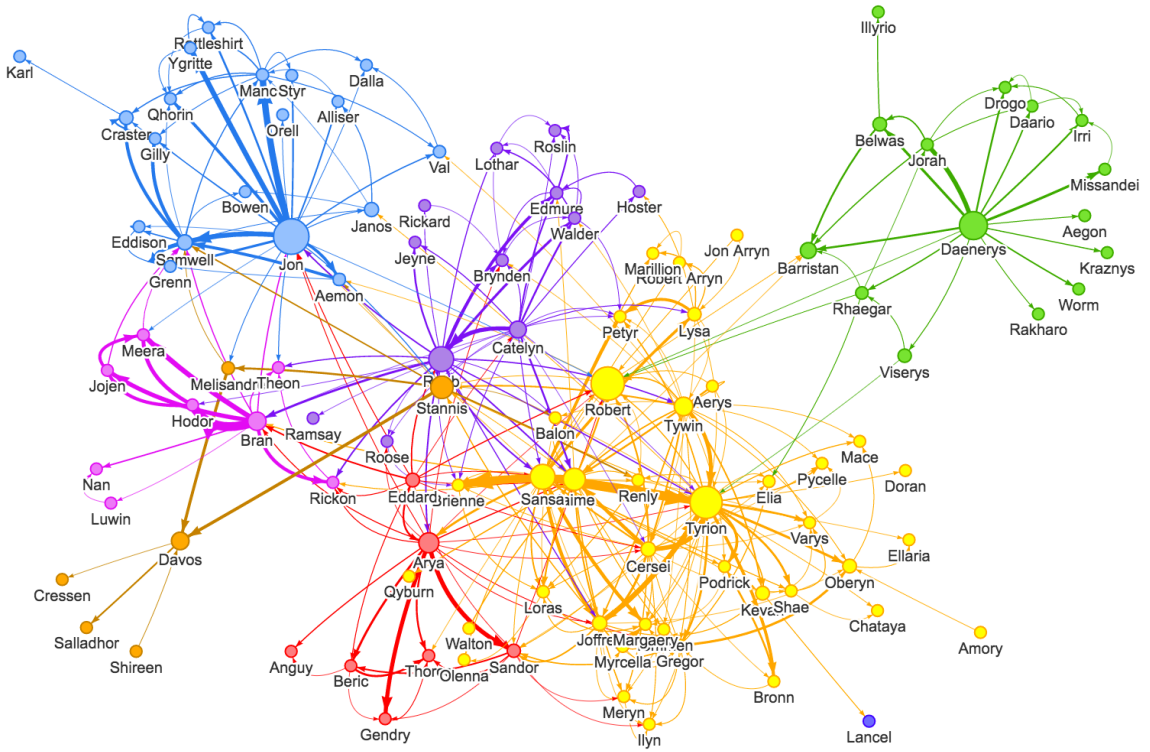
- **Clustering**
- **Centrality**
- **Similarity**

Use Case: Social Media Analysis

Lab: NetworkX Tutorial

Graph2Vec

Deep Learning on Graphs



https://siliconangle.com/2018/09/20/neo4j-tunes-graph-engine-ai-applications/

Motivation

Hochschule Offenburg
offenburg.university

Graphs are a basic data structure for many applications

Social Media

- **Who knows whom?**
- **Degree of connections**
- **Communication paths**
- **...**

http://digitalrezonance.com/the-social-graph

Motivation

Hochschule Offenburg
offenburg.university

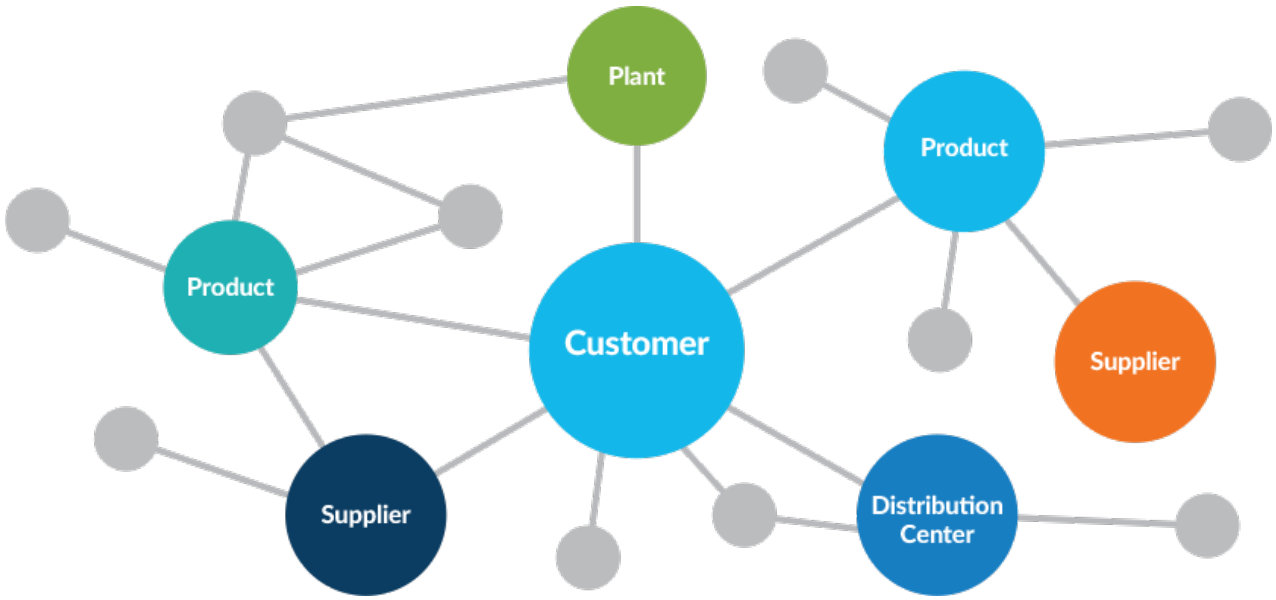
Graphs are a basic data structure for many applications

Logistics and supply chains

Product dependencies

Distribution/Delivery networks

...



http://fusionops.lnx.avisan.com/digital-cloud/

Motivation

Hochschule Offenburg
offenburg.university

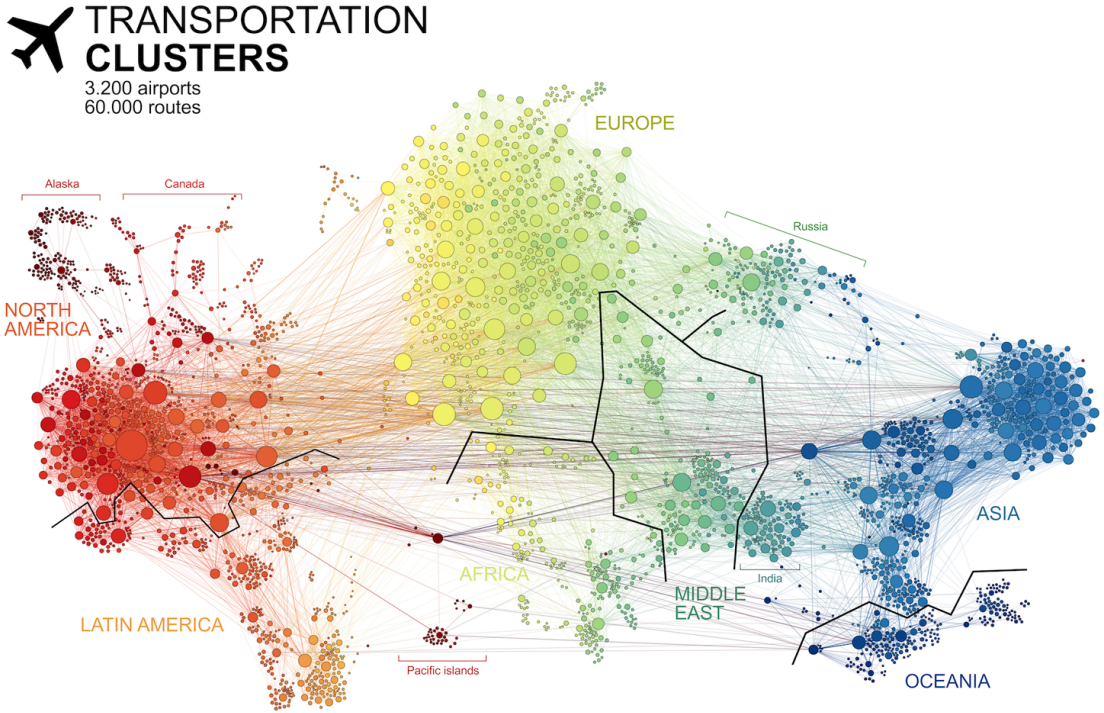
Graphs are a basic data structure for many applications

Travel analysis and Navigation

Routing

Traffic planing

...



https://www.oreilly.com/library/view/graph-algorithms/9781492047674/ch01.html

Graph Basics

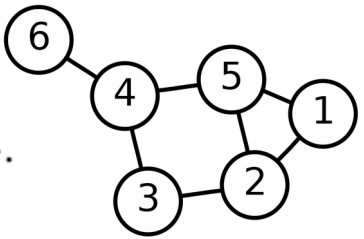
Hochschule Offenburg
offenburg.university

Recalling some basic graph properties

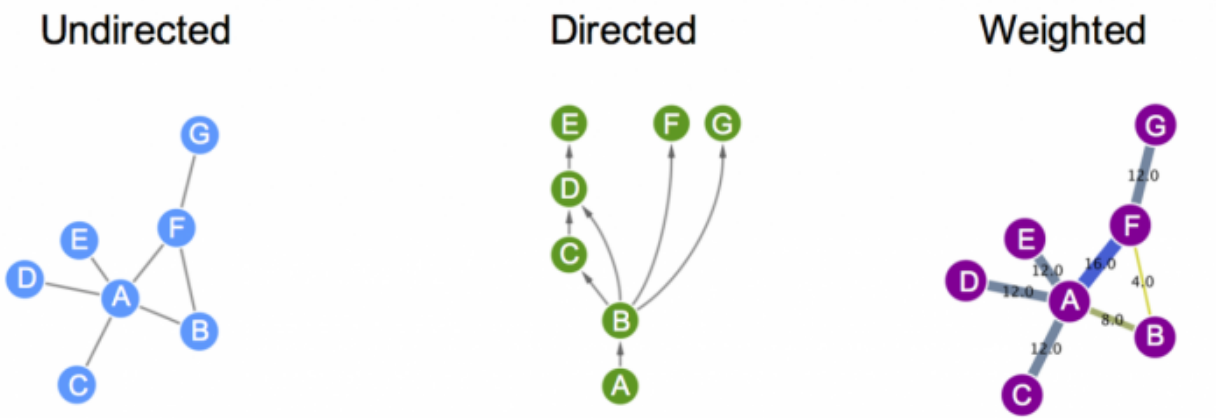
Definition $G = (E, V)$

with Edges $E = \{\{1,2\},\{1,5\},\{2,3\},\{2,5\},\{3,4\},\{4,5\},\{4,6\}\}.$

and Vertices $V = \{1,2,3,4,5,6\}$



Basic types



Graph figures from:
https://www.ebi.ac.uk

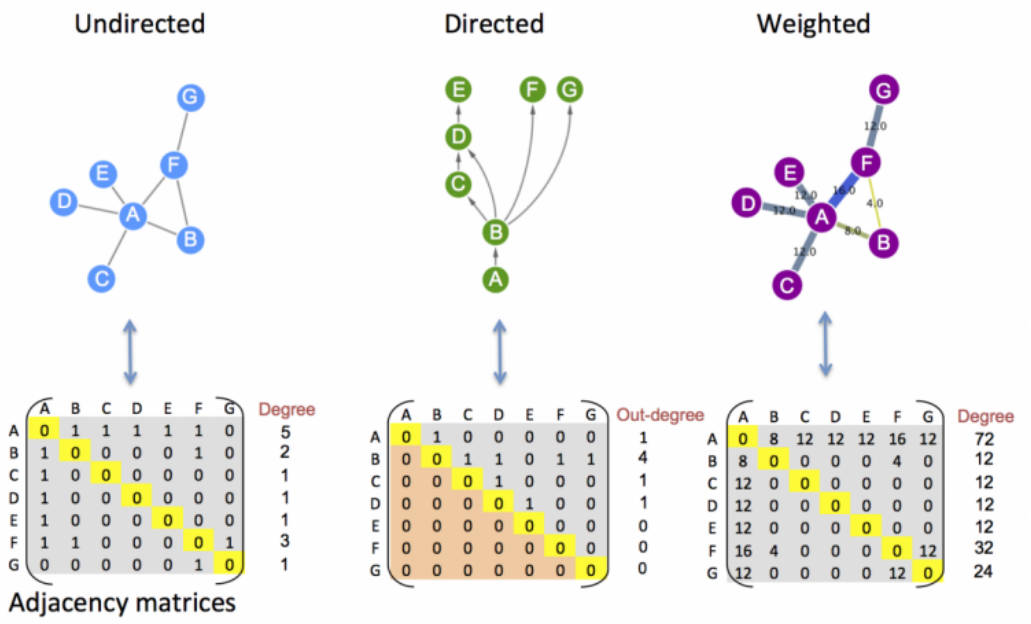
Graph Basics

Storing Graphs

Edge list

$$E = \{\{1,2\},\{1,5\},\{2,3\},\{2,5\},\{3,4\},\{4,5\},\{4,6\}\}.$$

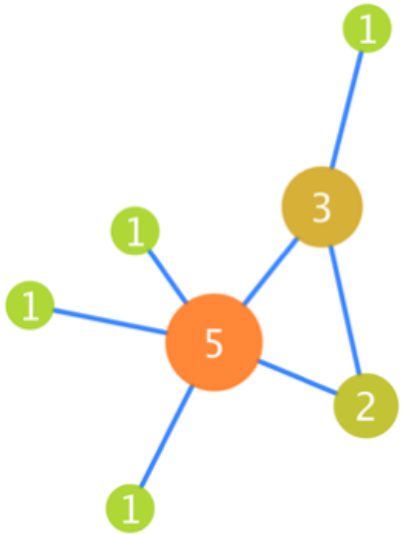
Adjacency Matrix



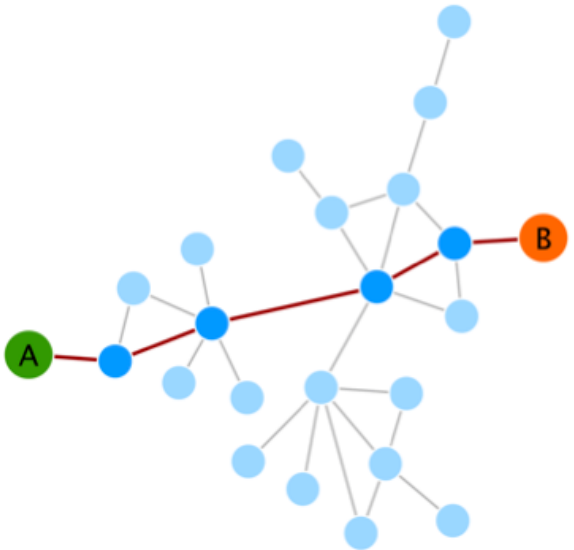
Graph Basics

Hochschule Offenburg
offenburg.university

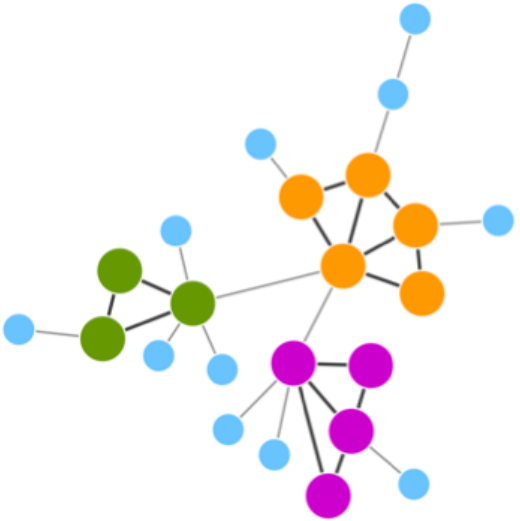
Topology Measures



Degree



(shortest) Paths



Clusters and Partition

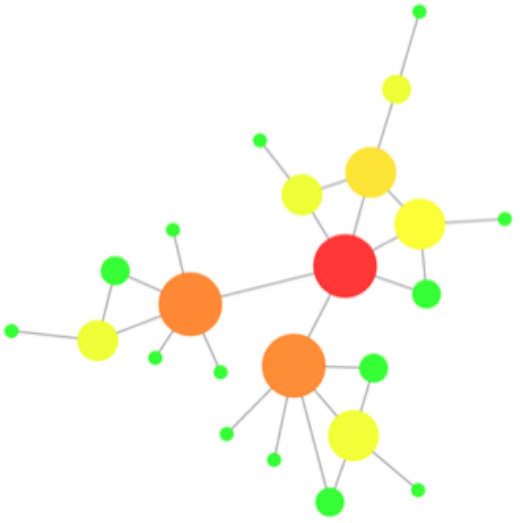
Graph Basics

Hochschule Offenburg
offenburg.university

Analysis: Centrality

There many formal definitions (depending on application and context)
Of centrality. However, the basic idea is the same:

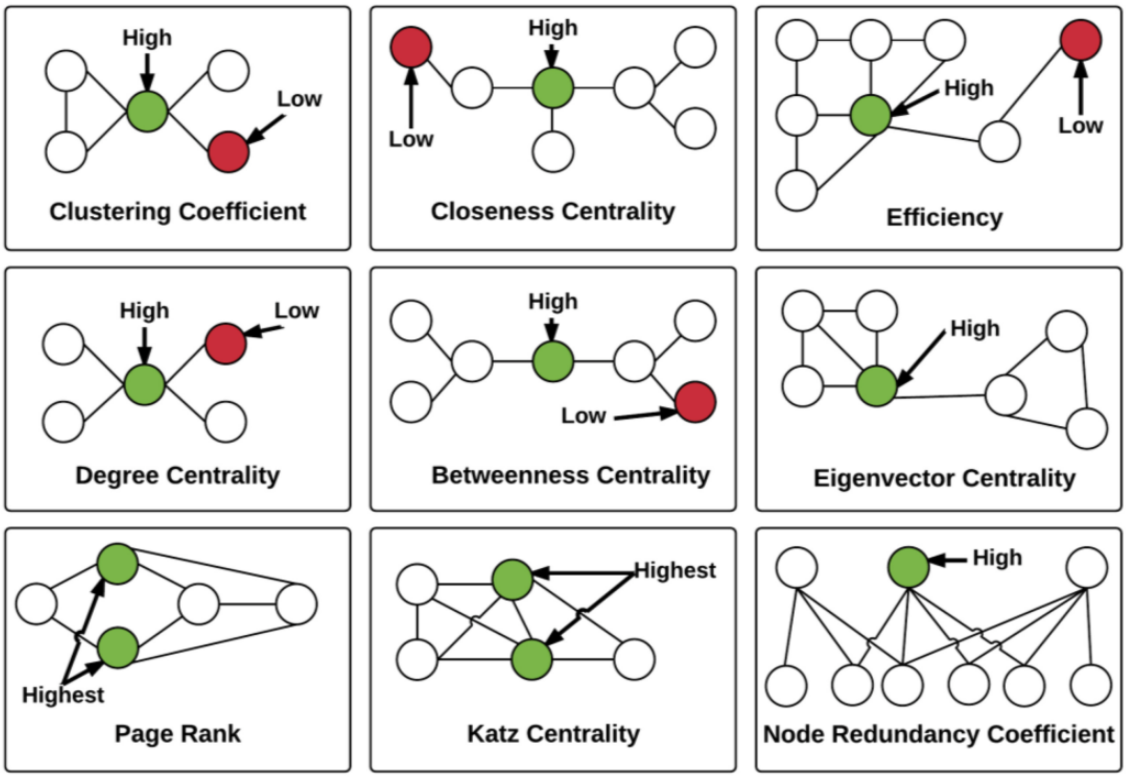
Find vertices that dominate (parts) of a graph.



Centrality

Centrality

Algorithm overview



Graphs in Python

Graph Libraries

There quite a few options, we will be using

→ **see lab tutorial**

Use Case

Hochschule Offenburg
offenburg.university

Facebook analysis: who are potential influencers ?

→ **Jupyter Notebook**

Graph2Vec

- **Unsupervised learning of graph embedding**
  - Graph → vector representation
  - Similar graphs are close in vector space
  - Pre-processing for clustering / classification
- **Follows the ideas of word2vec / doc2vec (text embedding)**
  - Using co-occurrence of words in sample text
  - Neural skip-gram

graph2vec: Learning Distributed Representations of Graphs

Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu and Shantanu Jaiswal
Nanyang Technological University, Singapore
annamala002@e.ntu.edu.sg,
{mahinthan,rajasekarv,elhchen,yangliu}@ntu.edu.sg,shantanu004@e.ntu.edu.sg

ABSTRACT

Recent works on representation learning for graph structured data predominantly focus on learning distributed representations of graph substructures such as nodes and subgraphs. However, many graph analytics tasks such as graph classification and clustering require representing entire graphs as fixed length feature vectors. While the aforementioned approaches are naturally unequipped to learn such representations, graph kernels remain as the most effective way of obtaining them. However, these graph kernels use handcrafted features (e.g., shortest paths, graphlets, etc.) and hence are hampered by problems such as poor generalization. To address this limitation, in this work, we propose a neural embedding framework named graph2vec to learn data-driven distributed representations of arbitrary sized graphs. graph2vec's embeddings are learnt in an unsupervised manner and are task agnostic. Hence, they could be used for any downstream task such as graph classification, clustering and even seeding supervised representation learning approaches. Our experiments on several benchmark and large real-world datasets show that graph2vec achieves significant improvements in classification and clustering accuracies over substructure representation learning approaches and are competitive with state-of-the-art graph kernels.

Keywords

Graph Kernels, Deep Learning, Representation Learning

1. INTRODUCTION

Graph-structured data are ubiquitous nowadays in many domains such as social networks, cybersecurity, bio- and chemo-informatics. Many analytics tasks in these domains such as graph classification, clustering and regression require representing graphs as fixed-length feature vectors to facilitate applying appropriate Machine Learning (ML) algorithms. For instance, vectorial representations (*aka* embeddings) of programs' call graphs could be used to detect malware [6] and those of chemical compounds could be used to predict their properties such as solubility and anti-cancer activity [7].

Graph Kernels and handcrafted features. Graph kernels are one of the most prominent ways of catering the aforementioned graph analytics tasks. Graph kernels evaluate the similarity (*aka* kernel value) between a pair of graphs G and G' by recursively decomposing them into atomic substructures (e.g., random walks, shortest paths, graphlets etc.) and defining a similarity (*aka* kernel) function over the substructures (e.g., counting the number of common substructures across G and G'). Subsequently, kernel methods (e.g., Support Vector Machines (SVMs)) could be used for performing classification/clustering. However, these kernels exhibit two critical limitations: (1) Many of them do not provide explicit graph embeddings. This renders using general purpose ML algorithms which operate on vector embeddings (e.g., Random Forests (RFs), Neural Networks, etc.) unusable with graph data. (2) The substructures (i.e., walk, paths, etc.) leveraged by these kernels are 'handcrafted' i.e., determined manually with specific well-defined functions that help extracting such substructures from graphs. However, as noted by Yanardag and Vishwanathan [7], when such handcrafted features are used on large datasets of graphs, it leads to building very high dimensional, sparse and non-smooth representations and thus yield poor generalization. We note that replacing handcrafted features with ones that are learnt automatically from data could help to fix both the aforementioned limitations. In fact, in related domains such as text mining and computer vision, feature learning based approaches have outperformed handcrafted ones significantly across many tasks [2, 9].

Learning substructure embeddings. Recently, several approaches have been proposed to learn embeddings of graph substructures such as nodes [4], paths [7] and subgraphs [5, 6]. These embeddings can then be used directly in substructure based analytics tasks such as node classification, community detection and link prediction. However, these substructure representation learning approaches are incapable of learning representations of entire graphs and hence cannot be used for tasks such as graph classification. As we show through our experiments in §5, obtaining graph embeddings through trivial extensions such as averaging or max pooling over substructure embeddings leads to suboptimal results.

Learning task-specific graph embeddings. On the other hand, a few supervised approaches (i.e., ones that require class labels of graphs) to learn embeddings of entire

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

Graph2Vec

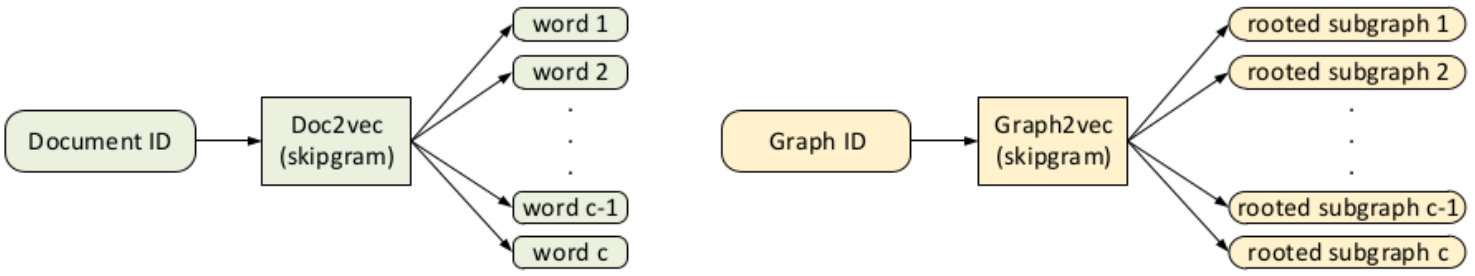
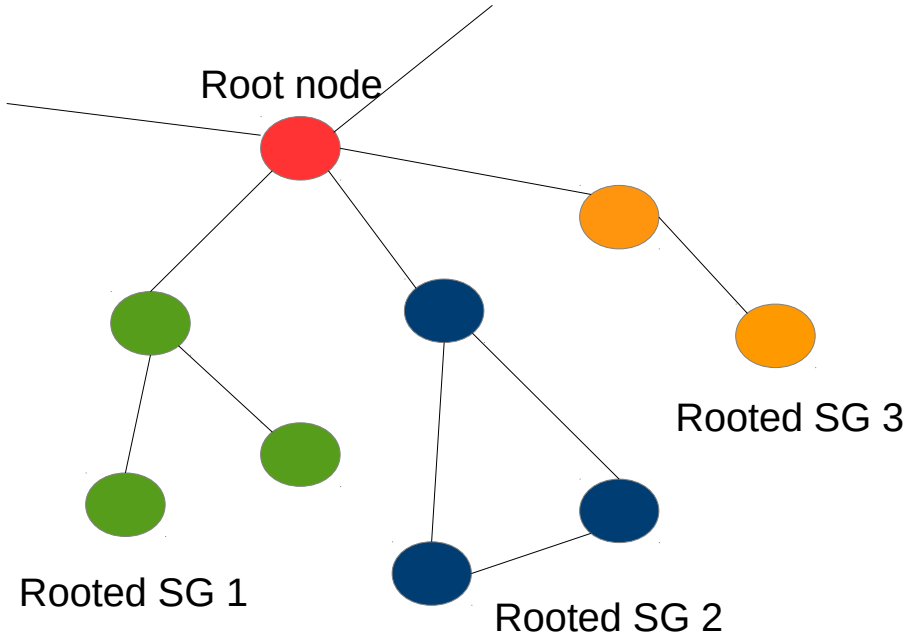


Figure 1: (a) `doc2vec`'s skipgram model - Given a document d, it samples c words from d and considers them as co-occurring in the same context (i.e., context of the document d) uses them to learn d's representation. (b) `graph2vec` - Given a graph G, it samples c rooted subgraphs around different nodes that occur in G and uses them analogous to `doc2vec`'s context words and thus learns G's representation.

"rooted subgraph" = word

Graph2Vec



Algorithm 2: GETWLSUBGRAPH (n, G, d)

input : n: Node which acts as the root of the subgraph
$G = (N, E, \lambda)$: Graph from which subgraph has to be extracted
d: Degree of neighbours to be considered for extracting subgraph

output: $sg_n^{(d)}$: Rooted subgraph of degree d around node n

begin
  $sg_n^{(d)} = \{\}$
  if $d = 0$ **then**
    $sg_n^{(d)} := \lambda(n)$
  else
    $\mathcal{N}_n := \{n' \mid (n, n') \in E\}$
    $M_n^{(d)} := \{\text{GETWLSUBGRAPH}(n', G, d-1) \mid n' \in \mathcal{N}_n\}$
    $sg_n^{(d)} := sg_n^{(d)} \cup$ GETWLSUBGRAPH $(n, G, d-1) \oplus sort(M_n^{(d)})$
  return $sg_n^{(d)}$

SG of size n: pre modeled look up table

Graph2Vec

Hochschule Offenburg
offenburg.university

Evaluation:

Table 2: Average Accuracy (± std dev.) for `graph2vec` and state-of-the-art graph kernels on benchmark graph classification datasets

Dataset	MUTAG	PTC	PROTEINS	NCI1	NCI109
`node2vec` [4]	72.63 ± 10.20	58.85 ± 8.00	57.49 ± 3.57	54.89 ± 1.61	52.68 ± 1.56
`sub2vec` [5]	61.05 ± 15.79	59.99 ± 6.38	53.03 ± 5.55	52.84 ± 1.47	50.67 ± 1.50
WL kernel [10]	80.63 ± 3.07	56.91 ± 2.79	72.92 ± 0.56	80.01 ± 0.50	80.12 ± 0.34
Deep WL kernel [7]	82.95 ± 1.96	59.04 ± 1.09	**73.30** ± 0.82	**80.31** ± 0.46	**80.32** ± 0.33
`graph2vec`	**83.15** ± 9.25	**60.17** ± 6.86	**73.30** ± 2.05	73.22 ± 1.81	74.26 ± 1.47

GCNs

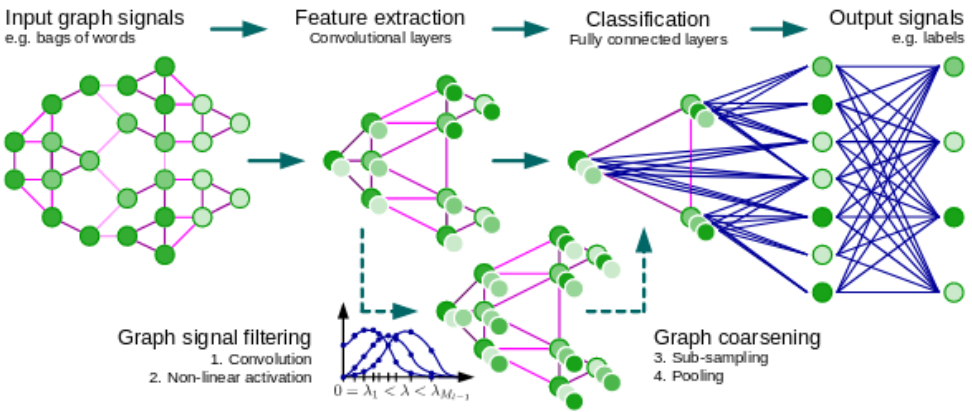


Figure 1: Architecture of a CNN on graphs and the four ingredients of a (graph) convolutional layer.

convolution

pooling

Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering

Michaël Defferrard **Xavier Bresson** **Pierre Vandergheynst**

EPFL, Lausanne, Switzerland

{michael.defferrard,xavier.bresson,pierre.vandergheynst}@epfl.ch

Abstract

In this work, we are interested in generalizing convolutional neural networks (CNNs) from low-dimensional regular grids, where image, video and speech are represented, to high-dimensional irregular domains, such as social networks, brain connectomes or words' embedding, represented by graphs. We present a formulation of CNNs in the context of spectral graph theory, which provides the necessary mathematical background and efficient numerical schemes to design fast localized convolutional filters on graphs. Importantly, the proposed technique offers the same linear computational complexity and constant learning complexity as classical CNNs, while being universal to any graph structure. Experiments on MNIST and 20NEWS demonstrate the ability of this novel deep learning system to learn local, stationary, and compositional features on graphs.

1 Introduction

Convolutional neural networks [19] offer an efficient architecture to extract highly meaningful statistical patterns in large-scale and high-dimensional datasets. The ability of CNNs to learn local stationary structures and compose them to form multi-scale hierarchical patterns has led to breakthroughs in image, video, and sound recognition tasks [18]. Precisely, CNNs extract the local stationarity property of the input data or signals by revealing local features that are shared across the data domain. These similar features are identified with localized convolutional filters or kernels, which are learned from the data. Convolutional filters are shift- or translation-invariant filters, meaning they are able to recognize identical features independently of their spatial locations. Localized kernels or compactly supported filters refer to filters that extract local features independently of the input data size, with a support size that can be much smaller than the input size.

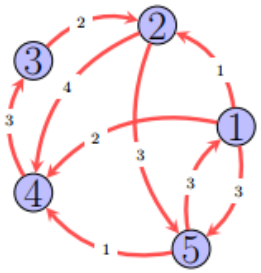
User data on social networks, gene data on biological regulatory networks, log data on telecommunication networks, or text documents on word embeddings are important examples of data lying on irregular or non-Euclidean domains that can be structured with graphs, which are universal representations of heterogeneous pairwise relationships. Graphs can encode complex geometric structures and can be studied with strong mathematical tools such as spectral graph theory [6].

A generalization of CNNs to graphs is not straightforward as the convolution and pooling operators are only defined for regular grids. This makes this extension challenging, both theoretically and implementation-wise. The major bottleneck of generalizing CNNs to graphs, and one of the primary goals of this work, is the definition of localized graph filters which are efficient to evaluate and learn. Precisely, the main contributions of this work are summarized below.

1. **Spectral formulation.** A spectral graph theoretical formulation of CNNs on graphs built on established tools in graph signal processing (GSP). [31].
2. **Strictly localized filters.** Enhancing [4], the proposed spectral filters are provable to be strictly localized in a ball of radius K, i.e. K hops from the central vertex.
3. **Low computational complexity.** The evaluation complexity of our filters is linear w.r.t. the filters support's size K and the number of edges $|\mathcal{E}|$. Importantly, as most real-world graphs are highly sparse, we have $|\mathcal{E}| \ll n^2$ and $|\mathcal{E}| = kn$ for the widespread k-nearest neighbor

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

GCNs: Convolution – Graph Fourier Transform



(a) An example directed graph.

“incoming edge weights from ...”

$$W = \begin{bmatrix} 0 & 0 & 0 & 0 & 3 \\ 1 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 2 & 4 & 0 & 0 & 1 \\ 3 & 3 & 0 & 0 & 0 \end{bmatrix}$$

(b) Weight matrix.

$$D_{in} = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 6 \end{bmatrix}$$

(c) In-degree matrix.

$$L = \begin{bmatrix} 3 & 0 & 0 & 0 & -3 \\ -1 & 3 & -2 & 0 & 0 \\ 0 & 0 & 3 & -3 & 0 \\ -2 & -4 & 0 & 7 & -1 \\ -3 & -3 & 0 & 0 & 6 \end{bmatrix}$$

(d) Laplacian matrix.

Fig. 1: A directed graph and the corresponding matrices.

“sum incoming edge weights”

Real, semi-positive definite

$L=D-W$

Ordered + real + positive Eigenvalues of L are
Used as the Fourier coefficients of a Graph

GCNs: Convolution – Graph Fourier Transform

Hochschule Offenburg
offenburg.university

Graph convolution: 1D convolution on the graph Fourier coefficients

→ **Problem:** GFC is a global operation, the most important property of CNNs is to encode local features ...

→ **Solution:** use localized (node-wise) sub-graph window to enforce local features

→ max distance from center node ~ filter size

GCNs: coarsening + pooling

Minimum cut coarsening

Pooling on 1D encoding

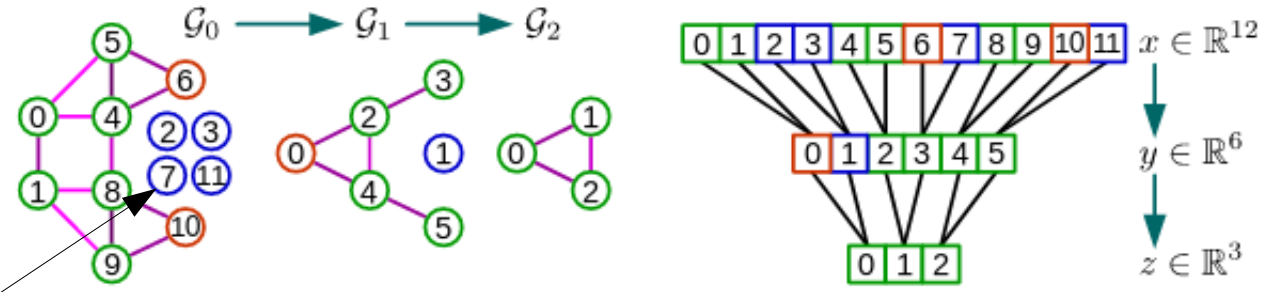


Figure 2: **Example of Graph Coarsening and Pooling.** Let us carry out a max pooling of size 4 (or two poolings of size 2) on a signal $x \in \mathbb{R}^8$ living on $\mathcal{G}_0$, the finest graph given as input. Note that it originally possesses $n_0 = |\mathcal{V}_0| = 8$ vertices, arbitrarily ordered. For a pooling of size 4, two coarsenings of size 2 are needed: let Graclus gives $\mathcal{G}_1$ of size $n_1 = |\mathcal{V}_1| = 5$, then $\mathcal{G}_2$ of size $n_2 = |\mathcal{V}_2| = 3$, the coarsest graph. Sizes are thus set to $n_2 = 3$, $n_1 = 6$, $n_0 = 12$ and fake nodes (in blue) are added to $\mathcal{V}_1$ (1 node) and $\mathcal{V}_0$ (4 nodes) to pair with the singeltons (in orange), such that each node has exactly two children. Nodes in $\mathcal{V}_2$ are then arbitrarily ordered and nodes in $\mathcal{V}_1$ and $\mathcal{V}_0$ are ordered consequently. At that point the arrangement of vertices in $\mathcal{V}_0$ permits a regular 1D pooling on $x \in \mathbb{R}^{12}$ such that $z = [\max(x_0, x_1), \max(x_4, x_5, x_6), \max(x_8, x_9, x_{10})] \in \mathbb{R}^3$, where the signal components x_2, x_3, x_7, x_{11} are set to a neutral value.

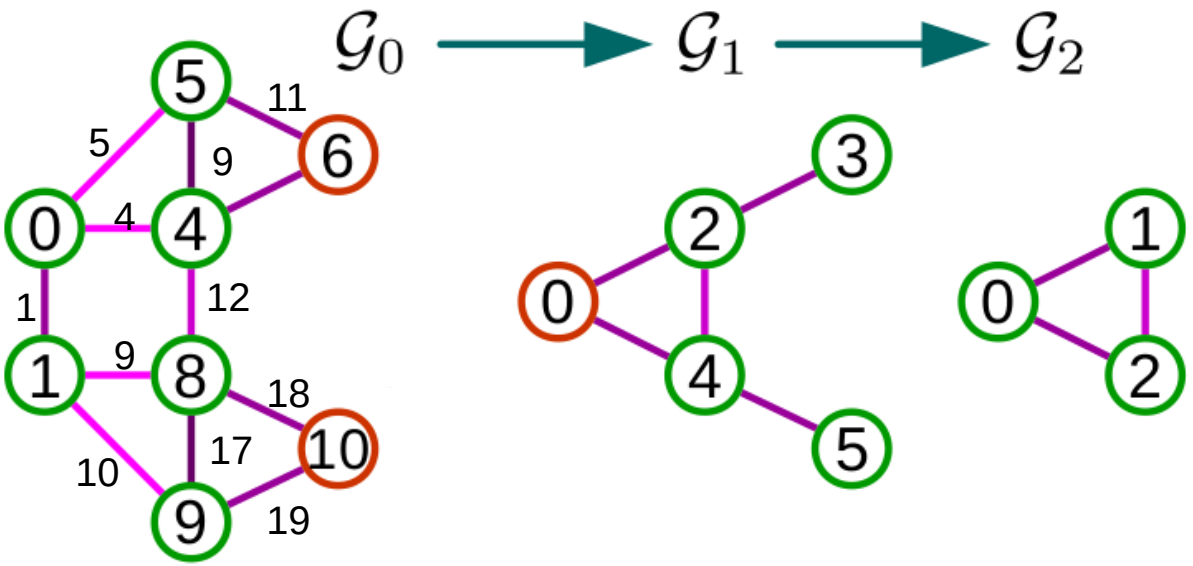
Dummy nodes:

Enforce min 2 children per node

Dummies are spectral neutral

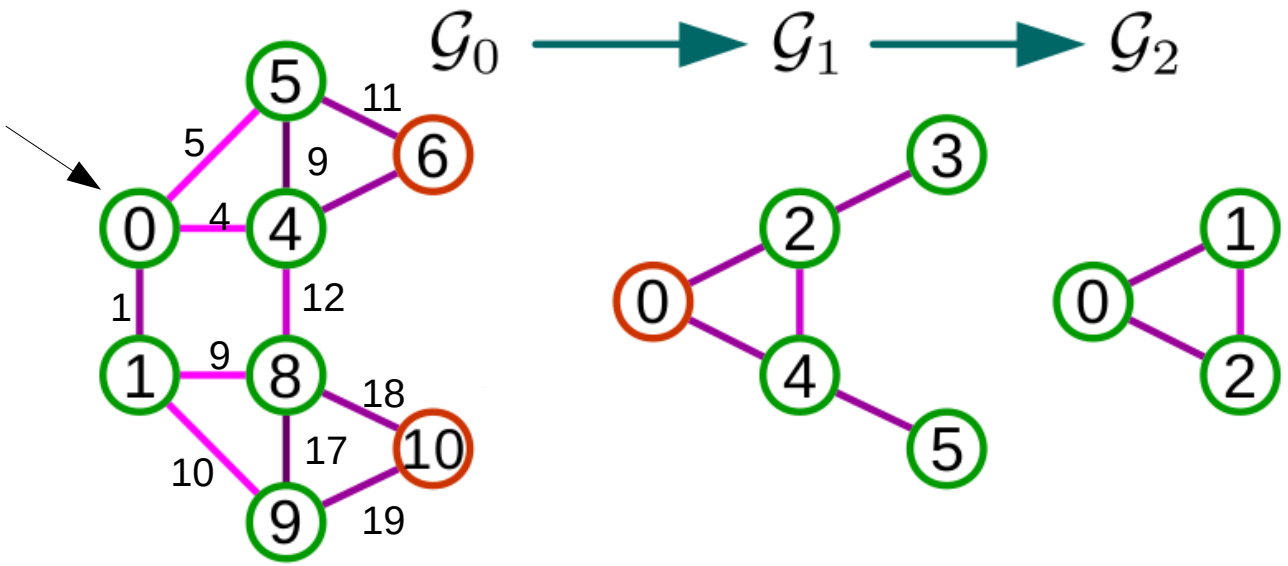
GCNs: coarsening + pooling

Minimum cut: Wij = di +dj



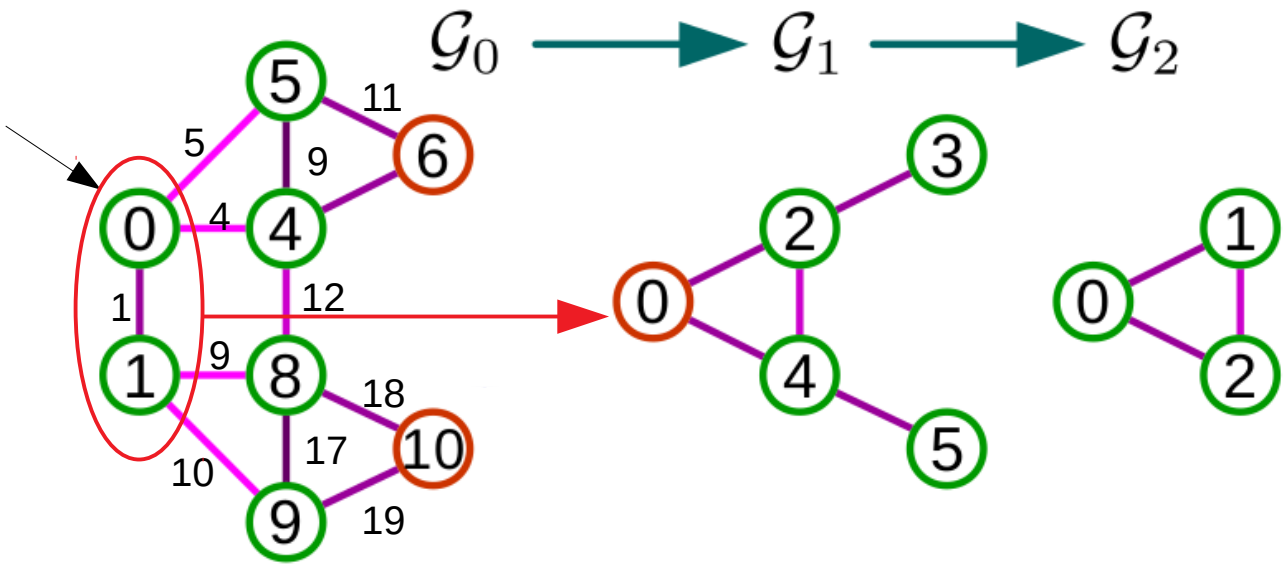
GCNs: coarsening + pooling

Minimum cut: Wij = di +dj



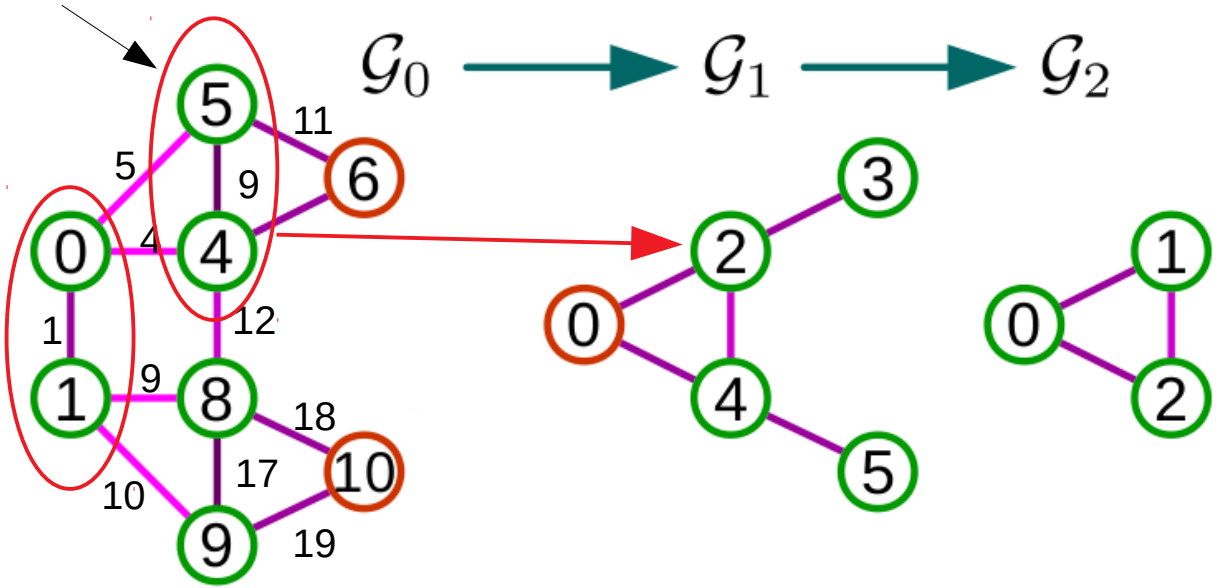
GCNs: coarsening + pooling

Minimum cut: Wij = di +dj



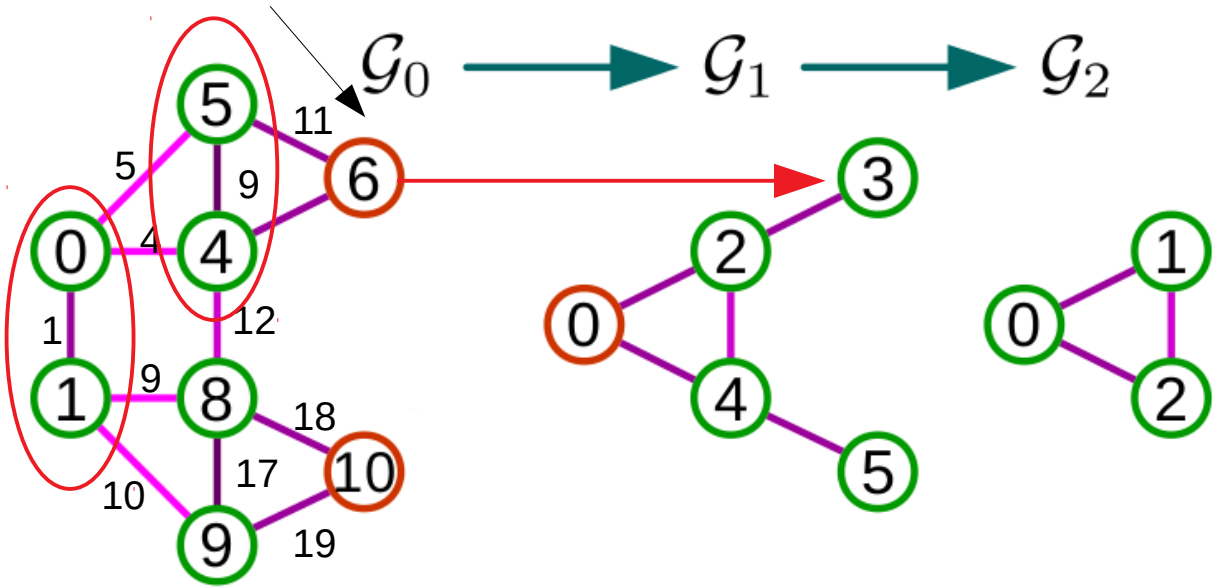
GCNs: coarsening + pooling

Minimum cut: Wij = di +dj



GCNs: coarsening + pooling

Minimum cut: Wij = di +dj



GCNs: coarsening + pooling

Minimum cut coarsening

Pooling on 1D encoding

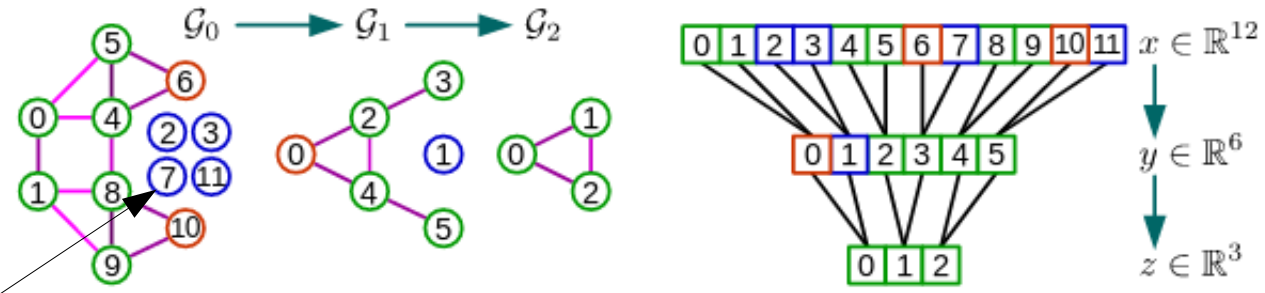


Figure 2: **Example of Graph Coarsening and Pooling.** Let us carry out a max pooling of size 4 (or two poolings of size 2) on a signal $x \in \mathbb{R}^8$ living on $\mathcal{G}_0$, the finest graph given as input. Note that it originally possesses $n_0 = |\mathcal{V}_0| = 8$ vertices, arbitrarily ordered. For a pooling of size 4, two coarsenings of size 2 are needed: let Graclus gives $\mathcal{G}_1$ of size $n_1 = |\mathcal{V}_1| = 5$, then $\mathcal{G}_2$ of size $n_2 = |\mathcal{V}_2| = 3$, the coarsest graph. Sizes are thus set to $n_2 = 3$, $n_1 = 6$, $n_0 = 12$ and fake nodes (in blue) are added to $\mathcal{V}_1$ (1 node) and $\mathcal{V}_0$ (4 nodes) to pair with the singeltons (in orange), such that each node has exactly two children. Nodes in $\mathcal{V}_2$ are then arbitrarily ordered and nodes in $\mathcal{V}_1$ and $\mathcal{V}_0$ are ordered consequently. At that point the arrangement of vertices in $\mathcal{V}_0$ permits a regular 1D pooling on $x \in \mathbb{R}^{12}$ such that $z = [\max(x_0, x_1), \max(x_4, x_5, x_6), \max(x_8, x_9, x_{10})] \in \mathbb{R}^3$, where the signal components x_2, x_3, x_7, x_{11} are set to a neutral value.

Dummy nodes:

Enforce min 2 children per node

Dummies are spectral neutral

GCNs: some Results

Hochschule Offenburg
offenburg.university

Prove of concept: MNIST images as graph*

* pixel values as node values

Model	Architecture	Accuracy
Classical CNN	C32-P4-C64-P4-FC512	99.33
Proposed graph CNN	GC32-P4-GC64-P4-FC512	99.14

Table 1: Classification accuracies of the proposed graph CNN and a classical CNN on MNIST.

Model	Architecture	CPU	GPU	Speedup
Classical CNN	C32-P4-C64-P4-FC512	210	31	6.77x
Proposed graph CNN	GC32-P4-GC64-P4-FC512	1600	200	8.00x

Table 4: Time to process a mini-batch of $S = 100$ MNIST images.

Compute complexity factor >~6

Discussion