CSI5180. Machine Learning for Bioinformatics Applications

Learning **Graphs**

by

Marcel Turcotte

Version **December 6, 2019**

Preamble

Preamble

Learning Graphs

A **graph** is a fundamental data structure with a great number of applications, both in computer science and the life sciences. In this lecture, we consider machine learning algorithms where graphs are playing a central role.

General objective :

- **Discuss** the applications of **frequent subgraph mining** in bioinformatics

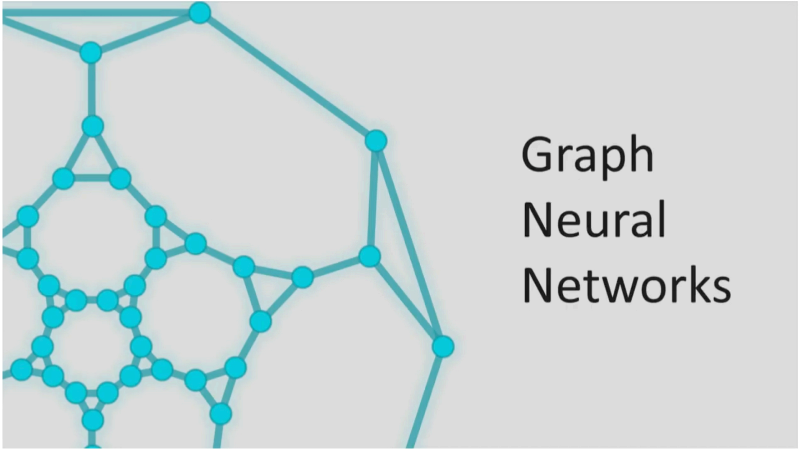
Learning objectives

- **Discuss** the various search strategies from frequent subgraph mining
- **Explain** the two main paradigms, single graph vs multiple graphs

Reading:

- Aida Mrzic, Pieter Meysman, Wout Bittremieux, Pieter Moris, Boris Cule, Bart Goethals, and Kris Laukens. Grasping frequent subgraph mining for bioinformatics applications. *BioData Min* **11**:20, 2018.
- Peng Zhang and Yuval Itan. Biological network approaches and applications in rare disease studies. *Genes* **10**: 2019.
- Hiroshi Mamitsuka. *Textbook of Machine Learning and Data Mining: with Bioinformatics Applications*. Global Data Science Publishing, 2018.
  - § 6, 7 and 8.

Graph neural networks by Alexander Gaunt



https://youtu.be/cWIeTMklzNg

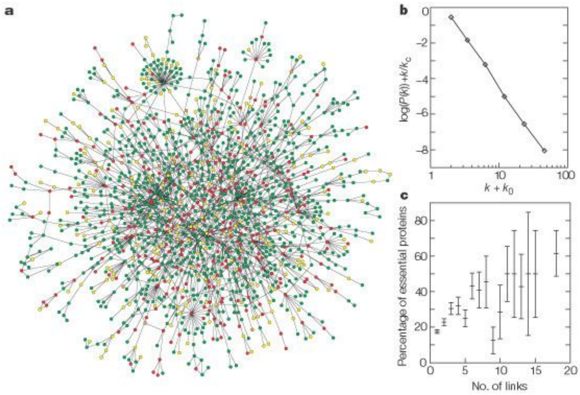
Plan

Introduction

Graphs in molecular biology

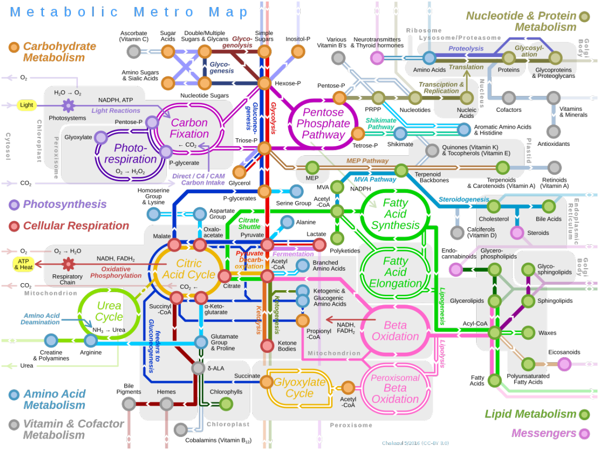
- Gene Regulatory Networks (GRN)
- Biological Pathways
- Protein-Protein Interactions (PPI)
- RNA-RNA Interaction (RRI)
- RNA secondary structure (tree, dual graph)
- Molecular graph (connectivity of molecules)
  - PubChem from NIH has 90 million entries
- Genome assembly
- Ontologies

Yeast proteome



H. Jeong, S. P. Mason, A.-L. Barabási & Z. N. Oltvai. Lethality and centrality in protein networks *Nature* **411**:4142 (2001)

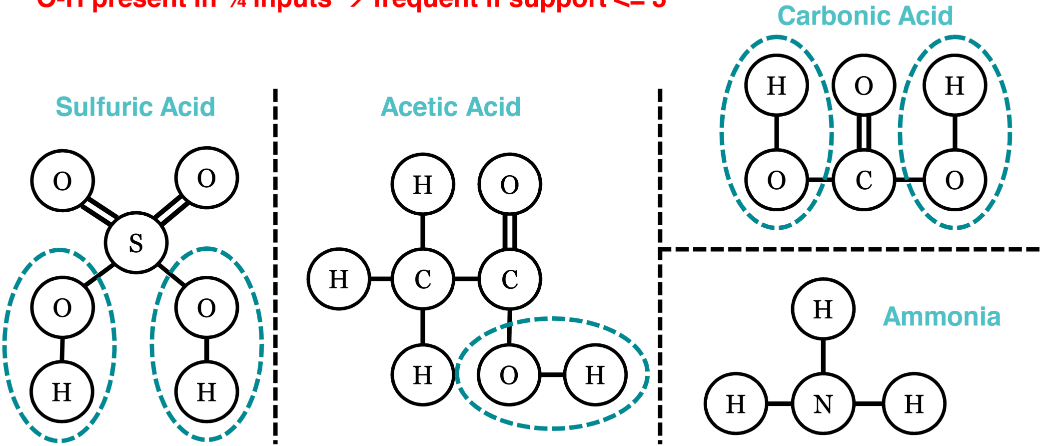
Metabolic network



Source: `https://en.wikipedia.org/wiki/File:Metabolic_Metro_Map.svg`

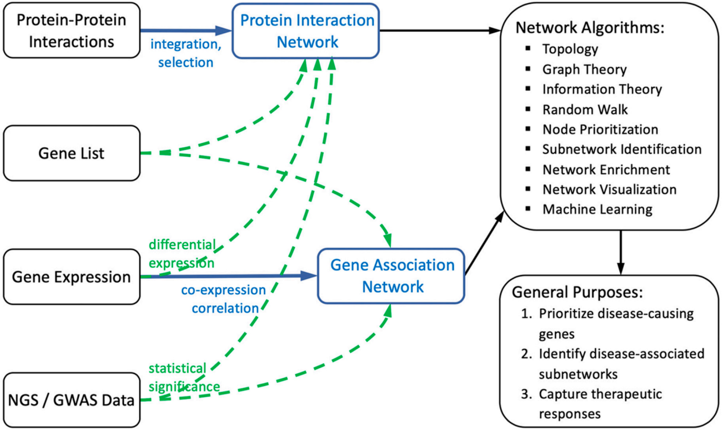
Molecular graph

O-H present in ¾ inputs → frequent if support <= 3



Source: [Samatova et al., 2013]

Biological networks and rare disease

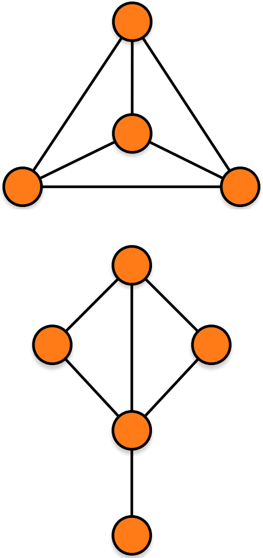


Source: [Zhang and Itan, 2019] Figure 1

Definitions

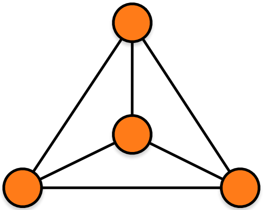
Graph

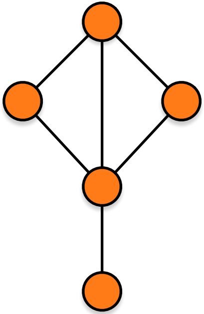
- A graph, $G = (V, E)$, consists of a set of **vertices** (V) and a set of **edges** (E) where each edge connects two nodes.



Graph

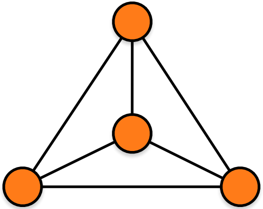
- A graph, $G = (V, E)$, consists of a set of **vertices** (V) and a set of **edges** (E) where each edge connects two nodes.
- A graph can be **labelled** or **unlabelled**. Both, edges and nodes can be labelled.

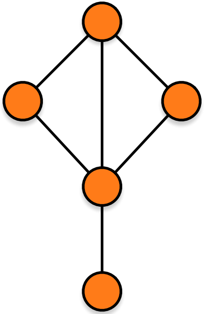




Graph

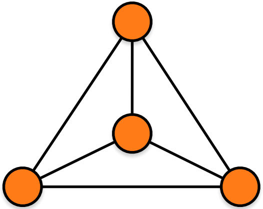
- A graph, $G = (V, E)$, consists of a set of **vertices** (V) and a set of **edges** (E) where each edge connects two nodes.
- A graph can be **labelled** or **unlabelled**. Both, edges and nodes can be labelled.
- An **edge** can be **directed** or **not**.

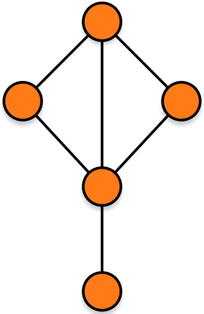




Graph

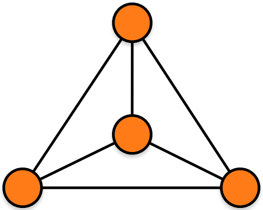
- A graph, $G = (V, E)$, consists of a set of **vertices** (V) and a set of **edges** (E) where each edge connects two nodes.
- A graph can be **labelled** or **unlabelled**. Both, edges and nodes can be labelled.
- An **edge** can be **directed** or **not**.
- There can be **weights** on **edges**. If so, the result is a **weighted graph**. Otherwise, the graph is **unweighted**.

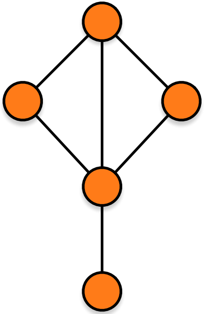




Graph

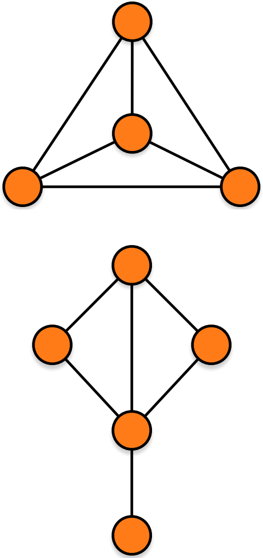
- **Nodes** are biological entities, such **atoms**, **molecules**, or **genes**.





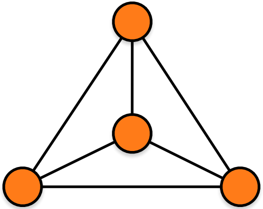
Graph

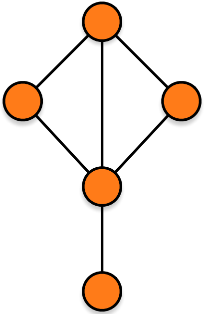
- **Nodes** are biological entities, such **atoms**, **molecules**, or **genes**.
- An **edge** represents an "association". For instance, a **chemical bond**, an **interaction**, or a **relationship** (e.g. regulates the activity of).



Graph

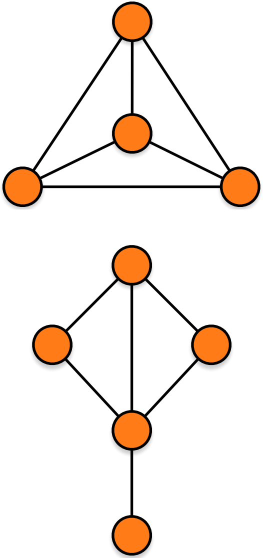
- **Nodes** are biological entities, such **atoms**, **molecules**, or **genes**.
- An **edge** represents an "association". For instance, a **chemical bond**, an **interaction**, or a **relationship** (e.g. regulates the activity of).
- **Weights** can be used describe a degree of **certainty** (e.g. experimental error) or **strength** of an association.





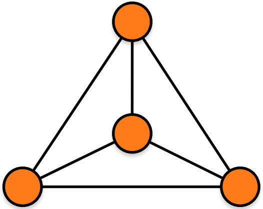
Subgraph

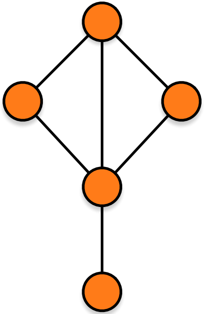
- A graph G_s is a **subgraph** of G if all the edges and nodes of G_s are subsets of the edges and nodes of G.



Subgraph

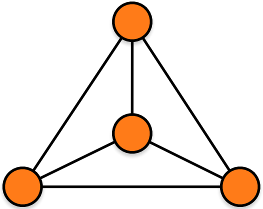
- A graph G_s is a **subgraph** of G if all the edges and nodes of G_s are subsets of the edges and nodes of G.
- A graph G_s is an **induced subgraph** of G, if the nodes of G_s are a subset of the nodes of G, and the nodes in G_s are connected <u>if and only if</u> they are connected in G.

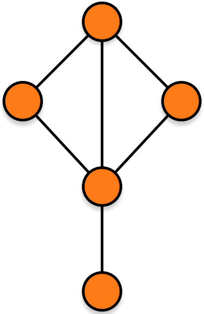




Subgraph

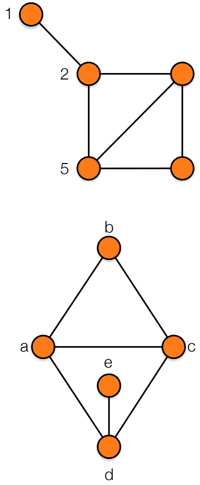
- A graph G_s is a **subgraph** of G if all the edges and nodes of G_s are subsets of the edges and nodes of G.
- A graph G_s is an **induced subgraph** of G, if the nodes of G_s are a subset of the nodes of G, and the nodes in G_s are connected <u>if and only if</u> they are connected in G.
- Herein, we focus on **connected subgraphs** where all the nodes are connected.





Isomorphic

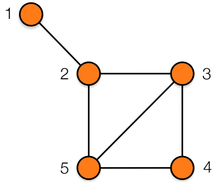
- Two graphs are **isomorphic** if there exists a **mapping** (bijection) between the **nodes** of the two graphs, such that **if two nodes are connected in one graph**, then **they are connected in the other**.

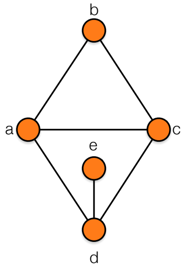


See also: `https://www.youtube.com/watch?v=Xq8o-z1DsUA`

Isomorphic

- Two graphs are **isomorphic** if there exists a **mapping** (bijection) between the **nodes** of the two graphs, such that **if two nodes are connected in one graph**, then **they are connected in the other**.
- In other words, the graphs can be seen as "**equal**".



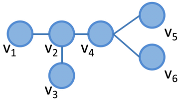


See also: https://www.youtube.com/watch?v=Xq8o-z1DsUA

Representations

Adjacency matrix and adjacency list

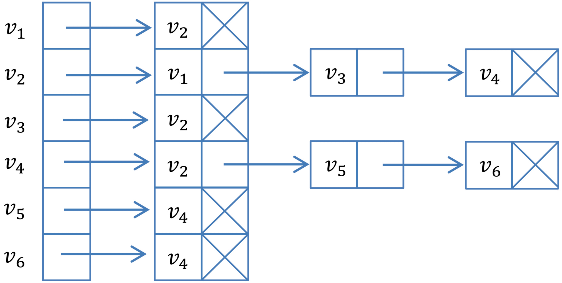
a



A graph G.

$$A(G) = \begin{array}{c c} & \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\ \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} & \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

b



Source: [Mrzic et al., 2018] Figure 3

Remark

- Two (2) **isomorphic** graphs do not <u>necessarily</u> have the same **adjacency matrix** or **adjacency list**!

Remark

- Two (2) **isomorphic** graphs do not necessarily have the same **adjacency matrix** or **adjacency list**!
- **Yikes!**

Canonical labelling

- The **canonical labelling** of a graph is a representation such that **if two graphs are isomorphic**, then **their canonical labelling is the same**.

Canonical labelling

- The **canonical labelling** of a graph is a representation such that **if two graphs are isomorphic**, then **their canonical labelling is the same**.
  - Here are two such **encodings**:

Canonical labelling

- The **canonical labelling** of a graph is a representation such that **if two graphs are isomorphic**, then **their canonical labelling is the same**.
  - Here are two such **encodings**:
    - **Canonical adjacency matrix (CAM)**

Canonical labelling

- The **canonical labelling** of a graph is a representation such that **if two graphs are isomorphic**, then **their canonical labelling is the same**.
  - Here are two such **encodings**:
    - **Canonical adjacency matrix (CAM)**
    - **Depth-first search (DFS) code**

Problems

Paradigms

Two **paradigms**:

- **Single graph:**

Paradigms

Two **paradigms**:

- **Single graph:**
  - For these applications, the **input** consists of a single graph, let's a Protein-Protein-Interaction network.

Paradigms

Two **paradigms**:

- **Single graph:**
  - For these applications, the **input** consists of a single graph, let's a Protein-Protein-Interaction network.
  - The **output** is a list of **frequently occurring subgraphs**.

Paradigms

Two **paradigms**:

- **Single graph:**
  - For these applications, the **input** consists of a single graph, let's a Protein-Protein-Interaction network.
  - The **output** is a list of **frequently occurring subgraphs**.
- **Multiple graphs:**

Paradigms

Two **paradigms**:

- **Single graph:**
  - For these applications, the **input** consists of a single graph, let's a Protein-Protein-Interaction network.
  - The **output** is a list of **frequently occurring subgraphs**.
- **Multiple graphs:**
  - For this class of problems, the **input** is a collection of graphs, for examples the connectivity of small compounds, all having a similar activity (e.g. HIV reverse-transcriptase inhibitors).

Paradigms

Two **paradigms**:

- **Single graph:**
  - For these applications, the **input** consists of a single graph, let's a Protein-Protein-Interaction network.
  - The **output** is a list of **frequently occurring subgraphs**.
- **Multiple graphs:**
  - For this class of problems, the **input** is a collection of graphs, for examples the connectivity of small compounds, all having a similar activity (e.g. HIV reverse-transcriptase inhibitors).
  - The **output** would be one or several sugraphs, each occurring in a large proportion (good support) of the input graphs.

Paradigms

Two **paradigms**:

- **Single graph:**
  - For these applications, the **input** consists of a single graph, let's a Protein-Protein-Interaction network.
  - The **output** is a list of **frequently occurring subgraphs**.
- **Multiple graphs:**
  - For this class of problems, the **input** is a collection of graphs, for examples the connectivity of small compounds, all having a similar activity (e.g. HIV reverse-transcriptase inhibitors).
  - The **output** would be one or several sugraphs, each occurring in a large proportion (good support) of the input graphs.
- The overarching theme is searching for **frequently occurring interesting subgraphs**.

Frequent subgraph mining

- **Input:** a **graph** (G) or **set of graphs** ($\mathcal{G}$).
- **Output:** **subgraphs** with good **support**.

$$\mathcal{F} = \{g | g \text{ is a subgraph of } G \text{ or } \mathcal{G}; \text{support}(g) \geq \text{minimum support}\}$$

  - Where **support** is a **problem specific** measure:
    - **Count** is larger than some threshold s.
    - **Statistical enrichment** compared to some background distribution.

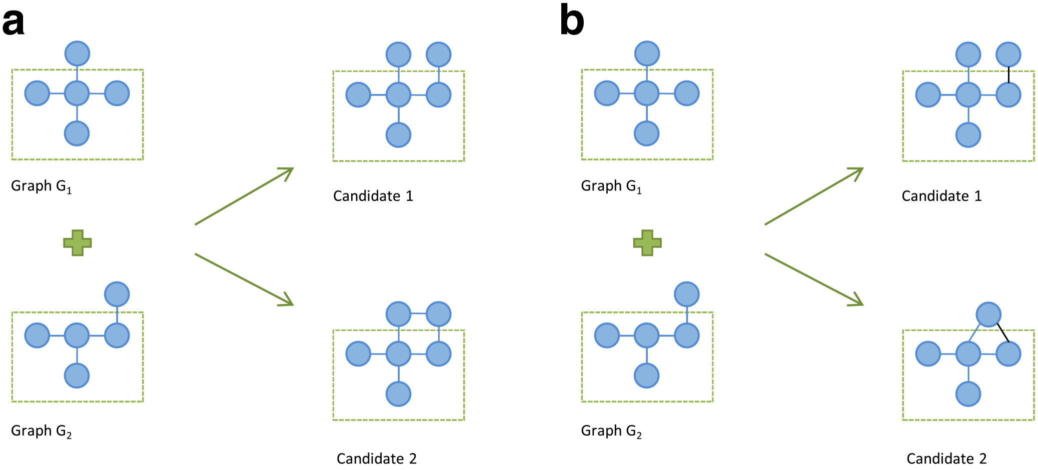
Algorithms

Frequent subgraph mining (high level)

1. **Enumerate** candidates
2. **Filter** the list
3. **Count** the number of occurrences
4. **Repeat**

Counting the number of occurrences is computationally demanding!

Join node- or edge-based enumeration



Source: [Mrzic et al., 2018] Figure 4

Pruning

- The candidate enumeration algorithms are joining subgraphs are are frequent [1].
  - The *a priori* principle says that **a graph cannot be more frequent than any of its subgraphs**.

[1] Count is higher than some threshold.

Search strategies

- **Breadth-first search (BFS) strategy**

Search strategies

- **Breadth-first search (BFS) strategy**
  - Candidates are enumerated in **order of size**.

Search strategies

- **Breadth-first search (BFS) strategy**
  - Candidates are enumerated in **order of size**.
    - Where **size** is either the **number of nodes** or **number of edges**.

Search strategies

- **Breadth-first search (BFS) strategy**
  - Candidates are enumerated in **order of size**.
    - Where **size** is either the **number of nodes** or **number of edges**.
  - **Cons:** large memory usage.

Search strategies

- **Breadth-first search (BFS) strategy**
  - Candidates are enumerated in **order of size**.
    - Where **size** is either the **number of nodes** or **number of edges**.
  - **Cons:** large memory usage.
- **Depth-first search (DFS) strategy**

Search strategies

- **Breadth-first search (BFS) strategy**
  - Candidates are enumerated in **order of size**.
    - Where **size** is either the **number of nodes** or **number of edges**.
  - **Cons:** large memory usage.
- **Depth-first search (DFS) strategy**
  - The algorithm keeps extending a candidate until the resulting subgraph is no longer frequent.

Search strategies

- **Breadth-first search (BFS) strategy**
  - Candidates are enumerated in **order of size**.
    - Where **size** is either the **number of nodes** or **number of edges**.
  - **Cons:** large memory usage.
- **Depth-first search (DFS) strategy**
  - The algorithm keeps extending a candidate until the resulting subgraph is no longer frequent.
  - **Cons:** pruning is less effective (thus large execution time).

Search strategies

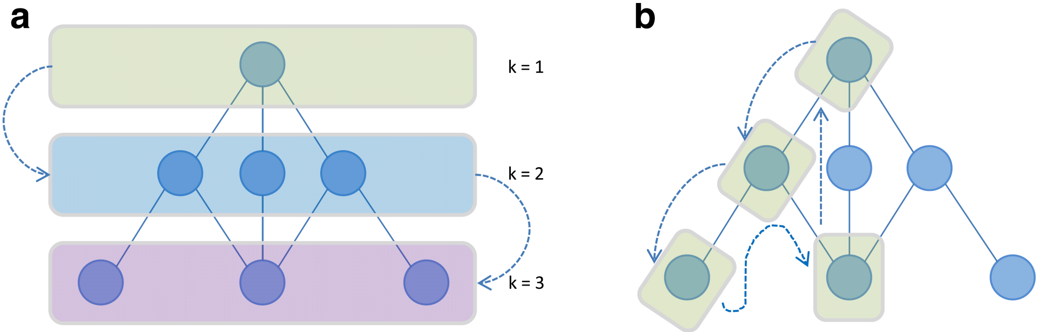
- **Breadth-first search (BFS) strategy**
  - Candidates are enumerated in **order of size**.
    - Where **size** is either the **number of nodes** or **number of edges**.
  - **Cons:** large memory usage.
- **Depth-first search (DFS) strategy**
  - The algorithm keeps extending a candidate until the resulting subgraph is no longer frequent.
  - **Cons:** pruning is less effective (thus large execution time).
- Use **all inducible subgraphs** (graphlets) up to a given size.

Search strategies

- **Breadth-first search (BFS) strategy**
  - Candidates are enumerated in **order of size**.
    - Where **size** is either the **number of nodes** or **number of edges**.
  - **Cons:** large memory usage.
- **Depth-first search (DFS) strategy**
  - The algorithm keeps extending a candidate until the resulting subgraph is no longer frequent.
  - **Cons:** pruning is less effective (thus large execution time).
- Use **all inducible subgraphs** (graphlets) up to a given size.
  - For instance, there are 30 undirected unlabelled connected inducible subgraphs of size 2 to 5.

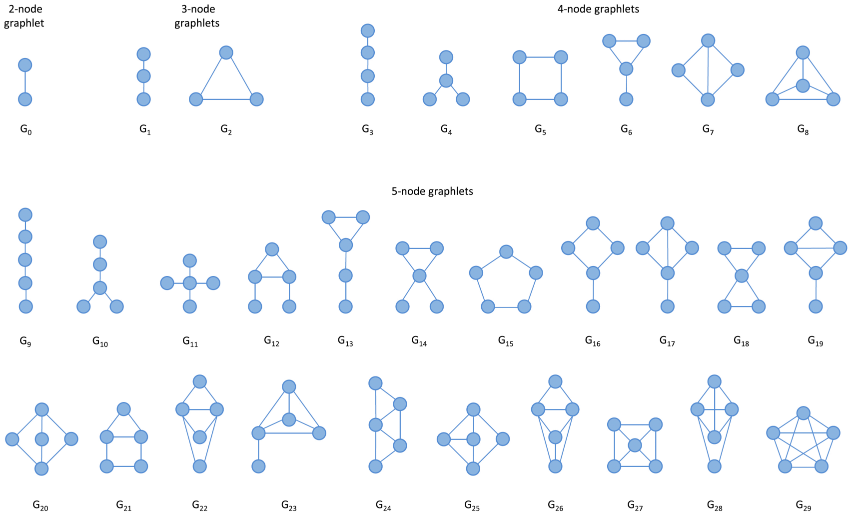
Strategies

p



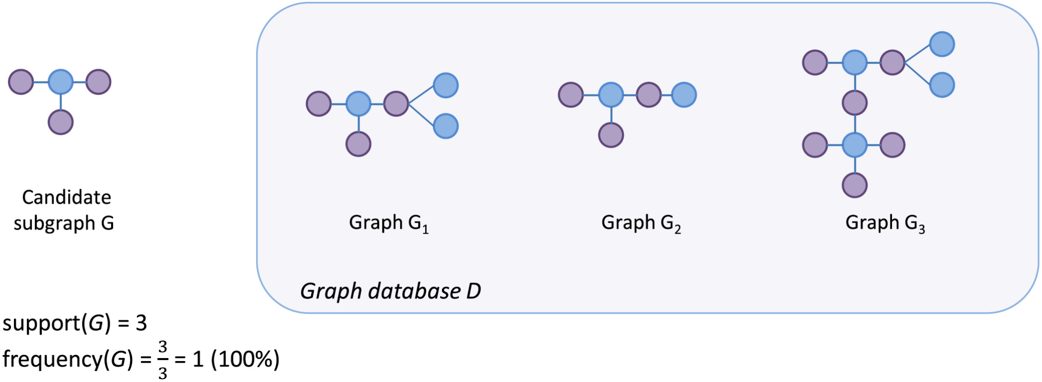
Source: [Mrzic et al., 2018] Figure 5

Strategies



Source: [Mrzic et al., 2018] Figure 6

Support (multiple graphs)



Source: [Mrzic et al., 2018] Figure 7

- When the input consists of **multiple graphs**, the support generally **ignores the number of times** a subgraph occurs in a given graph.

Support (single graph)

- Counting the number of occurrences in a single graph brings an **added level of complexity**.

Support (single graph)

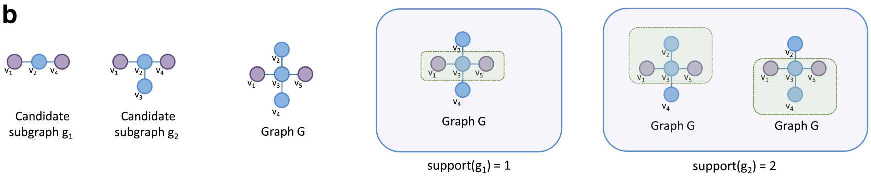
- Counting the number of occurrences in a single graph brings an **added level of complexity**.
  - Counting only the **non-overlapping** occurrences.

Support (single graph)

- Counting the number of occurrences in a single graph brings an **added level of complexity**.
  - Counting only the **non-overlapping** occurrences.
  - Counting all the occurrences, including the **overlapping** ones.

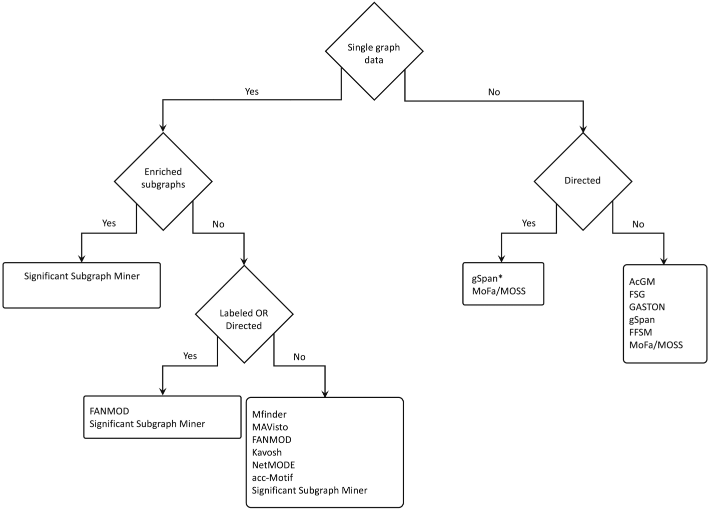
Support (single graph)

- Counting the number of occurrences in a single graph brings an **added level of complexity**.
  - Counting only the **non-overlapping** occurrences.
  - Counting all the occurrences, including the **overlapping** ones.
    - The *a priori* principle no longer applies as it is possible for larger subgraphs to occur more frequently than their subgraphs.



Source: [Mrzic et al., 2018] Figure 8

Existing approaches



Source: [Mrzic et al., 2018] Figure 9

Sampling

- In many cases, particularly in the case of a **single large graph**, an **exhaustive search is not feasible**.

Sampling

- In many cases, particularly in the case of a **single large graph**, an **exhaustive search is not feasible**.
  - **Sampling** approaches are then used.

Sampling

- In many cases, particularly in the case of a **single large graph**, an **exhaustive search is not feasible**.
  - **Sampling** approaches are then used.
  - **See:** Alex R Gawronski and Marcel Turcotte, RiboFSM: Frequent subgraph mining for the discovery of RNA structures and interactions, *BMC bioinformatics* (2014).

Prologue

Summary

- There are two paradigms, **single** graph and **multiple** graphs.

Summary

- There are two paradigms, **single** graph and **multiple** graphs.
- Frequent subgraph mining returns **all subgraphs** with **minimum support**.

Summary

- There are two paradigms, **single** graph and **multiple** graphs.
- Frequent subgraph mining returns **all subgraphs** with **minimum support**.
- Algorithms often proceed from **small to large subgraphs**, either using **breadth-first-search** or **depth-first-search**.

Summary

- There are two paradigms, **single** graph and **multiple** graphs.
- Frequent subgraph mining returns **all subgraphs** with **minimum support**.
- Algorithms often proceed from **small to large subgraphs**, either using **breadth-first-search** or **depth-first-search**.
- Depending on the application, the support can be the **count** or some **statistical test**.

Summary

- There are two paradigms, **single** graph and **multiple** graphs.
- Frequent subgraph mining returns **all subgraphs** with **minimum support**.
- Algorithms often proceed from **small to large subgraphs**, either using **breadth-first-search** or **depth-first-search**.
- Depending on the application, the support can be the **count** or some **statistical test**.
- When the graphs are **large**, **sampling** methods are used.

Resources

- **Graph Theory FAQs: 03. Isomorphism Using Adjacency Matrix** by **Sarada Herke**
  - https://youtu.be/UCle3Smvh1s
- **Graph Theory: 10. Isomorphic and Non-Isomorphic Graphs** by **Sarada Herke**
  - https://www.youtube.com/watch?v=z-GfKbzvtBA&feature=youtu.be

Next module

- **Ensemble** Learning

References

Gawronski, A. R. and Turcotte, M. (2014).
RiboFSM: Frequent subgraph mining for the discovery of RNA structures and interactions.
BMC bioinformatics.

Mamitsuka, H. (2018).
Textbook of Machine Learning and Data Mining: with Bioinformatics Applications.
Global Data Science Publishing.

Mrzic, A., Meysman, P., Bittremieux, W., Moris, P., Cule, B., Goethals, B., and Laukens, K. (2018).
Grasping frequent subgraph mining for bioinformatics applications.
BioData Min, 11:20.

Samatova, N., Hendrix, W., Jenkins, J., Padmanabhan, K., and Chakraborty, A. (2013).
Practical Graph Mining with R.
Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. CRC Press.

References

Zhang, P. and Itan, Y. (2019).
Biological network approaches and applications in rare disease studies.
Genes (Basel), 10(10).

Marcel **Turcotte**

Marcel.Turcotte@uOttawa.ca

School of Electrical Engineering and **Computer Science** (EECS)

University of Ottawa