Data Science with Graphs

— Graph Structure Analysis 2 —

Matteo Lissandrini – University of Verona

Outline

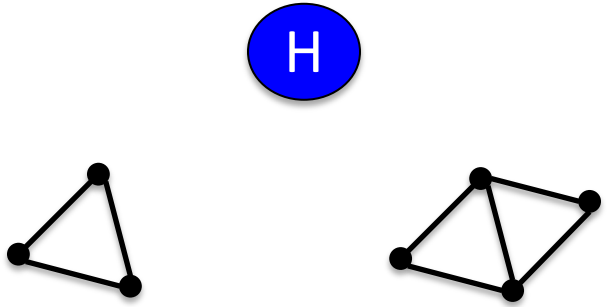
1. **Graph Structure Mining**
   - Graph homomorphism/isomorphism
   - Simulation/Strong Simulation
   - Graphlets/Motifs
   - Triangle Counting
2. **Frequent Graph Pattern Mining**
   - Graph-level subgraph frequency
   - Frequent Subgraph Mining
   - Extracting Shapes

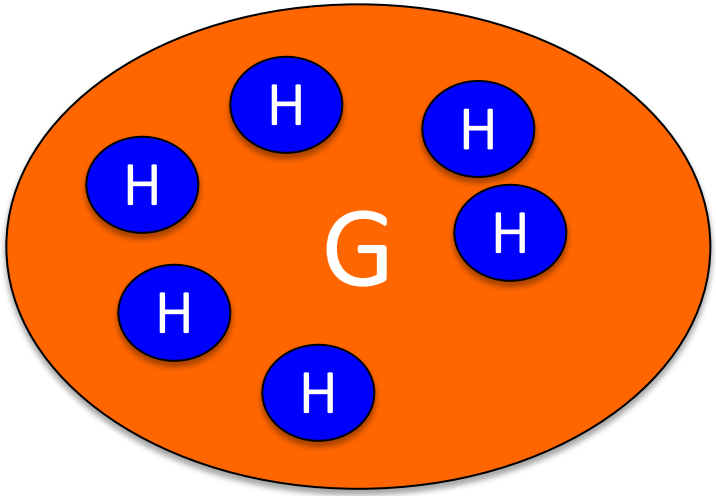
3. **Graph Clustering & Communities**
   - Connected components & Cliques
   - K-core & K-core decomposition
   - Community detection
   - Graph Partitioning & Conductance
   - Girvan Newman algorithm
   - Overlapping Communities
4. **Network Sampling**
   - Scale-down / Back-in-time sampling
   - Induced vs. Incident graph sampling
   - Biased Node Sampling
   - Shape Sampling

Graph Structure Mining

— Subgraph and Motif Search

Graph Structure Matching

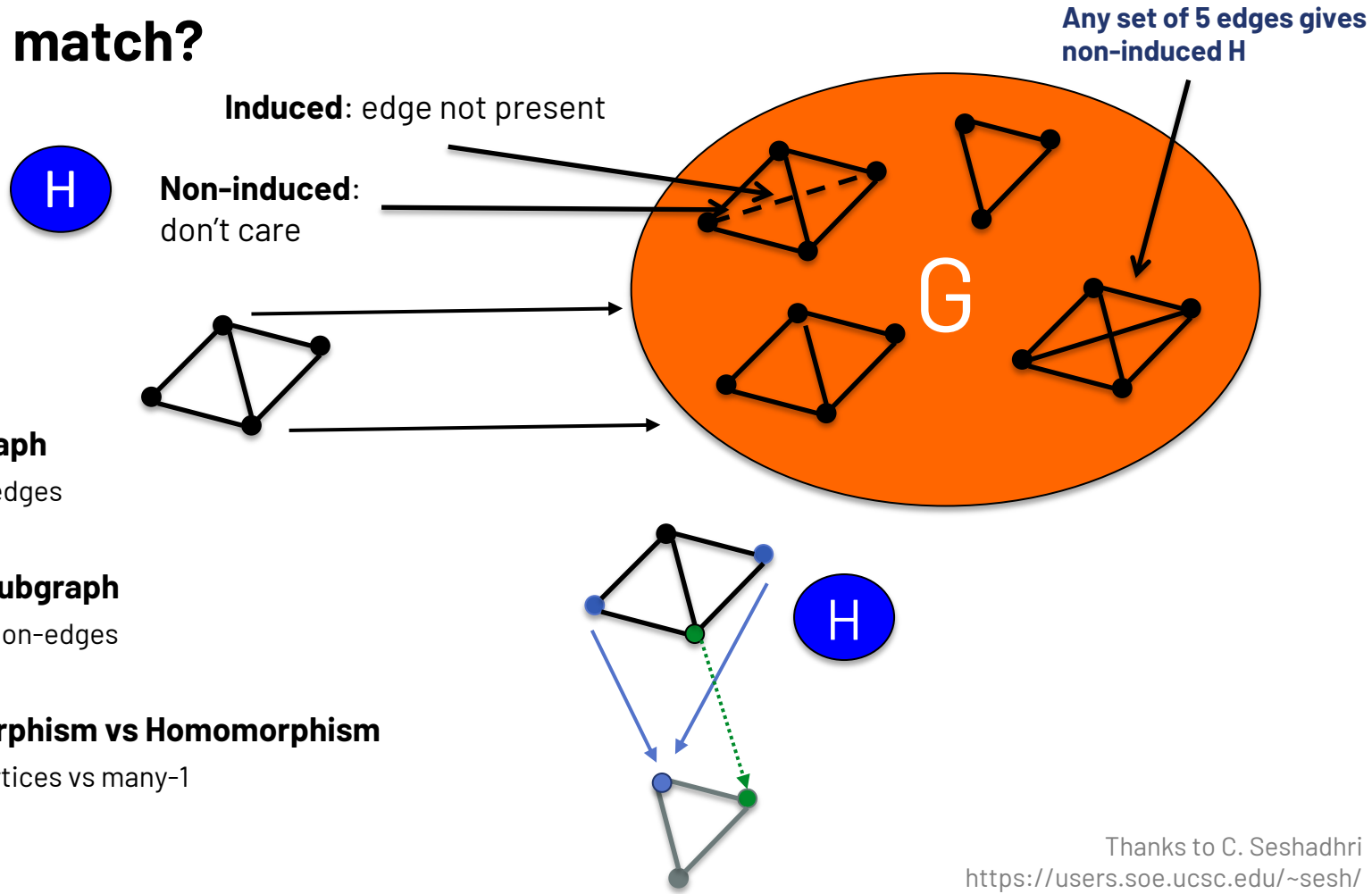




- G is a large graph (the input)
- H is a small "pattern" graph (the query)
- Count/find all occurrences of H in G
- Other names: graphlet analysis, motif counting

Thanks to C. Seshadhri
https://users.soe.ucsc.edu/~sesh/

What is a match?



Induced H-subgraph

- Must match non-edges

Non-induced H-subgraph

- Don't care about non-edges

Subgraph-isomorphism vs Homomorphism

- 1-1 mapping of vertices vs many-1

Thanks to C. Seshadhri
https://users.soe.ucsc.edu/~sesh/

Homomorphism vs. Isomorphism

An **isomorphism** between two graphs G and H is a **bijective mapping**

$$\phi : G \mapsto H$$

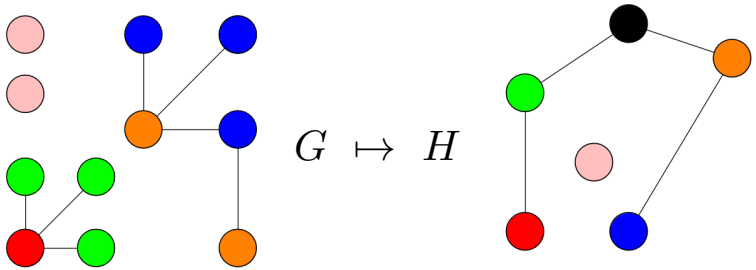
such that

$$(x, y) \in E(G) \Leftrightarrow (\phi(x), \phi(y)) \in E(H)$$

A **homomorphism** between two graphs G and H is a mapping

$$(x, y) \in E(G) \Rightarrow (\phi(x), \phi(y)) \in E(H)$$

Isomorphism is a stricter condition, while homomorphism just preserves edges in one direction



http://buzzard.ups.edu/courses/2013spring/projects/davis-homomorphism-ups-434-2013.pdf

Isomorphism vs. Subgraph-isomorphism

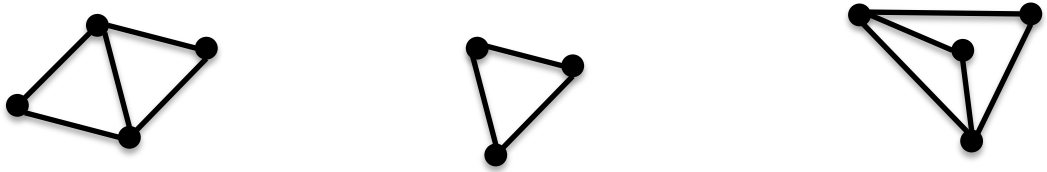
As decision problem:

- **Isomorphism** between G and H:
  is G *isomorphic to* H *?*

Subgraph-Isomorphism is NP-complete

Isomorphism is ¯_(ツ)_/¯

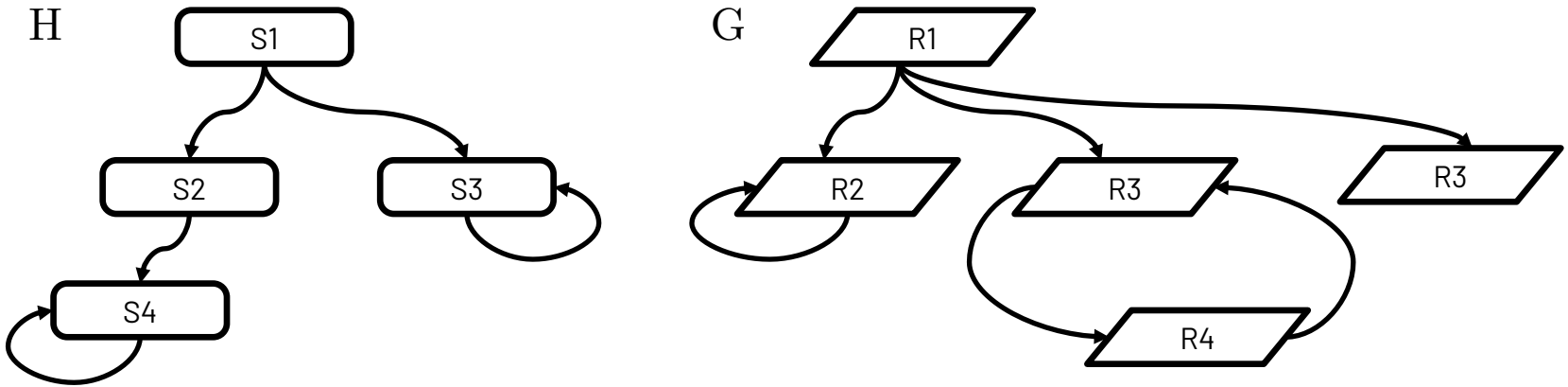
- **Subgraph-isomorphism** between G and H:
  is there a subgraph $H_0 \sqsubseteq H$ *such that* G *is isomorphic to* H_0



Complexity classes are tricky:

In reality, the subgraph isomorphism problem can be considered to be solvable in polynomial time in size of target graph if we consider that in practice the query graph has a bounded size!

Graph Simulation Matching



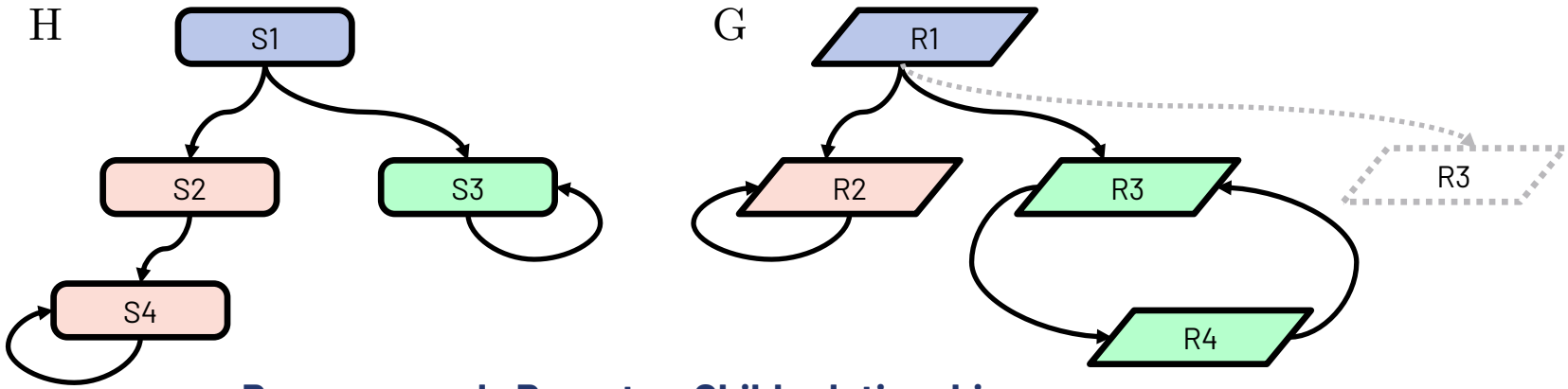
A graph G can simulate a graph H if there exists **a binary relation** $Sim \subseteq V_G \times V_H$ where for each $(u,v) \in Sim$ *if* $(v, v') \in E_H$ there is (u,u') in E_G such that $(u',v') \in Sim$

If I can follow one transition (edge) in one graph, the other should also be able to follow in the same way

https://dl.acm.org/doi/pdf/10.1145/2528937

Graph Simulation Matching

Simulation is computable in Polynomial time



Preserves only Parent → Child relationships

A graph G can simulate a graph H if there exists **a binary relation** $Sim \subseteq V_G \times V_H$ where for each $(u,v) \in Sim$ *if* $(v, v') \in E_H$ there is (u,u') in E_G such that $(u',v') \in Sim$

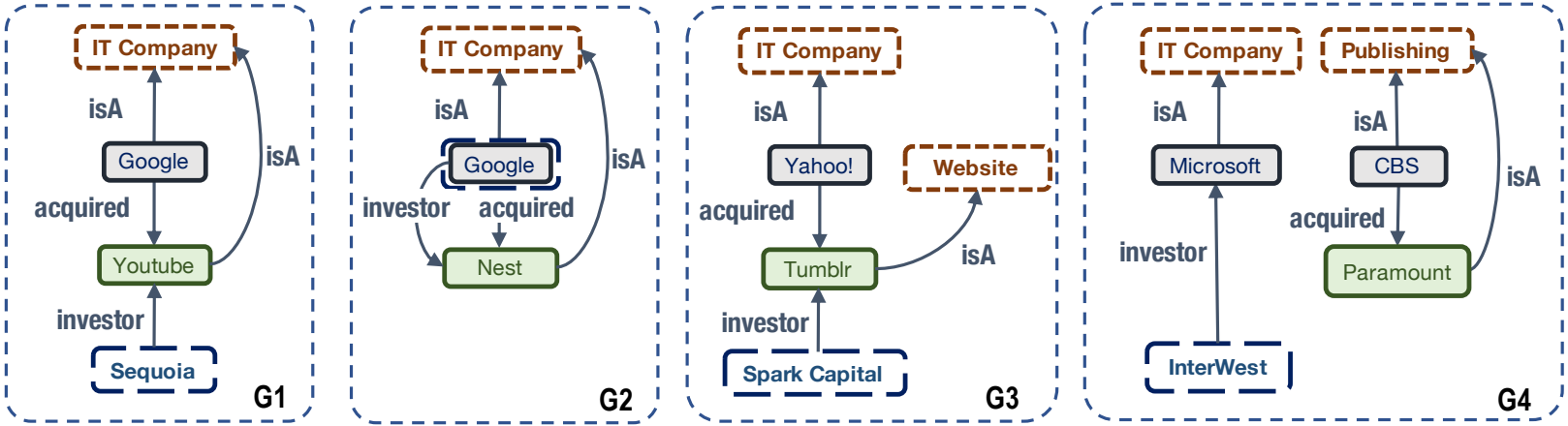
Usually, Simulation takes into consideration Node/Edge Labels which I've ignored in my example!

https://dl.acm.org/doi/pdf/10.1145/2528937

Graph Isomorphism vs. Simulation Variants

Structural Congruence/Similarity

Isomorphism requires an bijective **function**
Simulation requires only a **parent-child** edge preserving **relation**
Strong Simulation requires also **child-parent**, **connectivity** and **limited diameter**

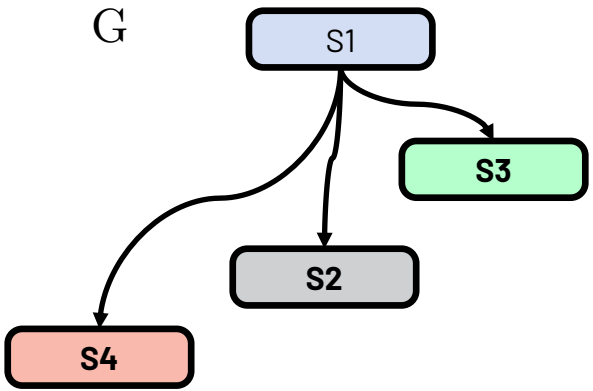


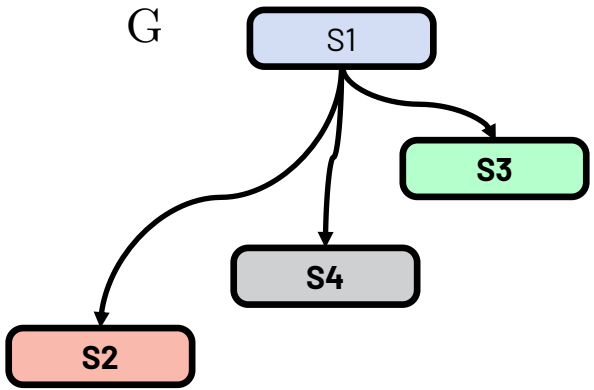
Example of Simulating (G1~ {G2,G3,G4}) and Strong-simulating Graphs (G1≈G2)

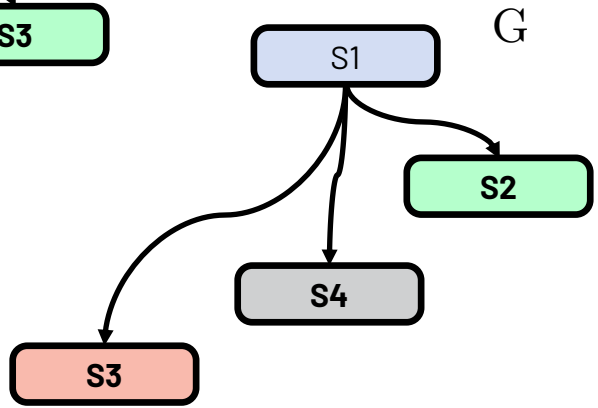
Strong Simulation preserves close connectivity

Strong simulation: Capturing topology in graph pattern matching – Shuai Ma et al., 2014

Automorphism







- A graph G is trivially isomorphic to itself
  - so it exists always at least one bijective mapping $\phi : G \mapsto G$
  - but what if there exist more than one $\phi_1 \neq \phi_2 \neq \ldots : G \mapsto G$?
- Automorphism: an homomorphism from a graph to itself

Challenge: *A Graph Structure Matching Algorithm needs to account for automorphism, and possibly avoid duplicate computation*

Analyzing Network Motifs

(*) Martorana, E., Micale, G., Ferro, A. et al. Establish the expected number of induced motifs on unlabeled graphs through analytical models. https://doi.org/10.1007/s41109-020-00294-y

- **Network Motif:** a recurrent, significant node-induced subgraph structure in a network (graph)
  - **Recurrent → Frequent:** How to identify "frequency"?
  - **Significant → appears more often than in a random graph**

Z_i is the **significance score** of S_i
negative value indicates
under-representation

$$Z_i = \frac{N_i^{\text{real}} - \bar{N}_i^{\text{rand}}}{\text{std}\left(N_i^{\text{rand}}\right)}$$

← Average frequency of S_i in random graphs (*)

- **Motifs analysis is based on network structure** (i.e., not considering labels)

What are the **possible motifs** that can appear in a graph given a fixed number of nodes K?

They are called **Graphlets**

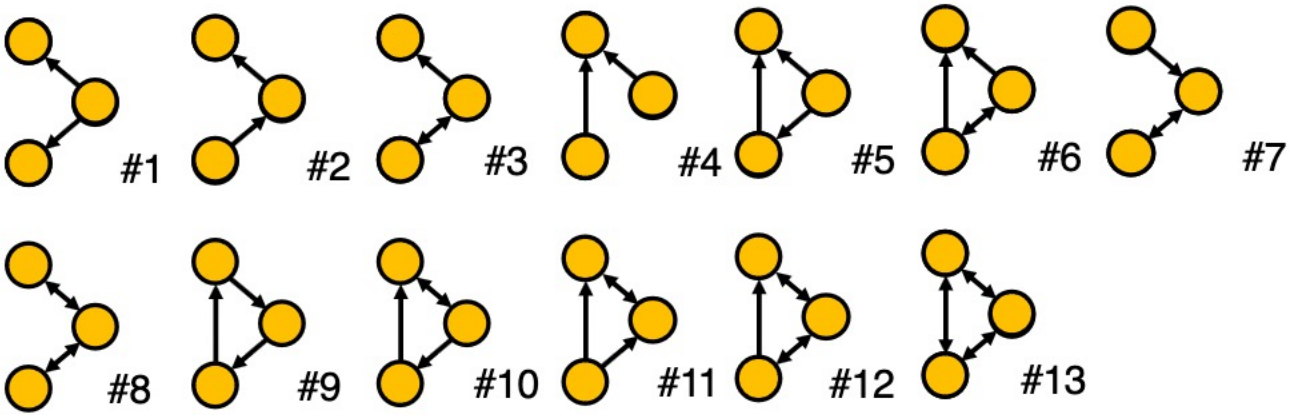
http://web.stanford.edu/class/cs224w/slides/12-motifs.pdf

Graphlets

A graphlet is a class of connected isomorphic subgraphs of a given size.

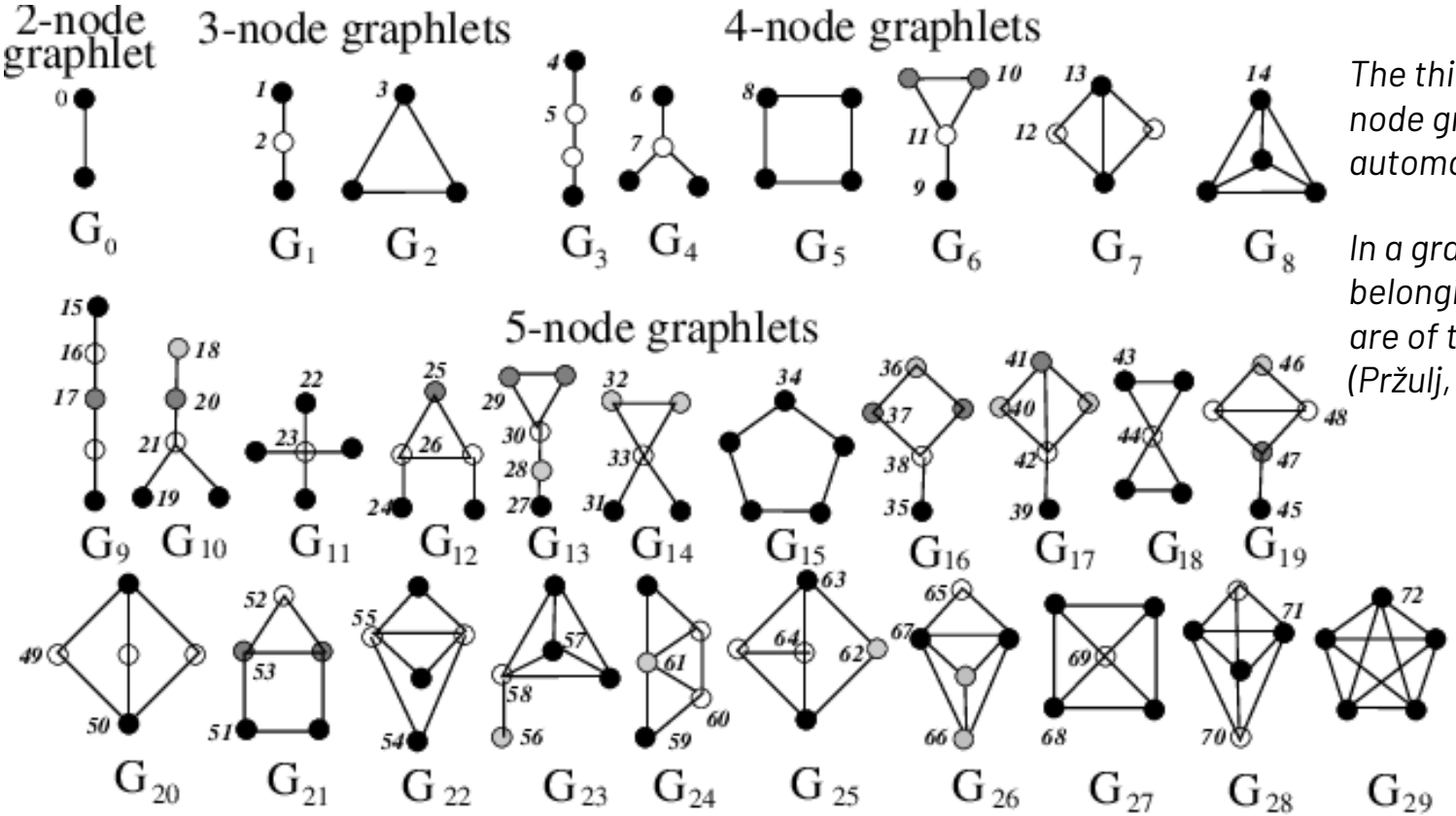
Two graphlet occurrences are isomorphic, whereas two occurrences of two distinct graphlets are non-isomorphic

Example: all non-isomorphic, connected, directed graphltes of size 3



http://web.stanford.edu/class/cs224w/slides/12-motifs.pdf

There are 30 Graphlets of size ⩽5 but ~11M of size 10



The thirty 2-, 3-, 4-, and 5-node graphlets and their automorphism orbits

In a graphlet G_i, nodes belonging to the same orbit are of the same shade (Pržulj, 2006).

Pržulj, Nataša. "Biological network comparison using graphlet degree distribution." *Bioinformatics* 23.2 (2006): e177-e183.

Outline

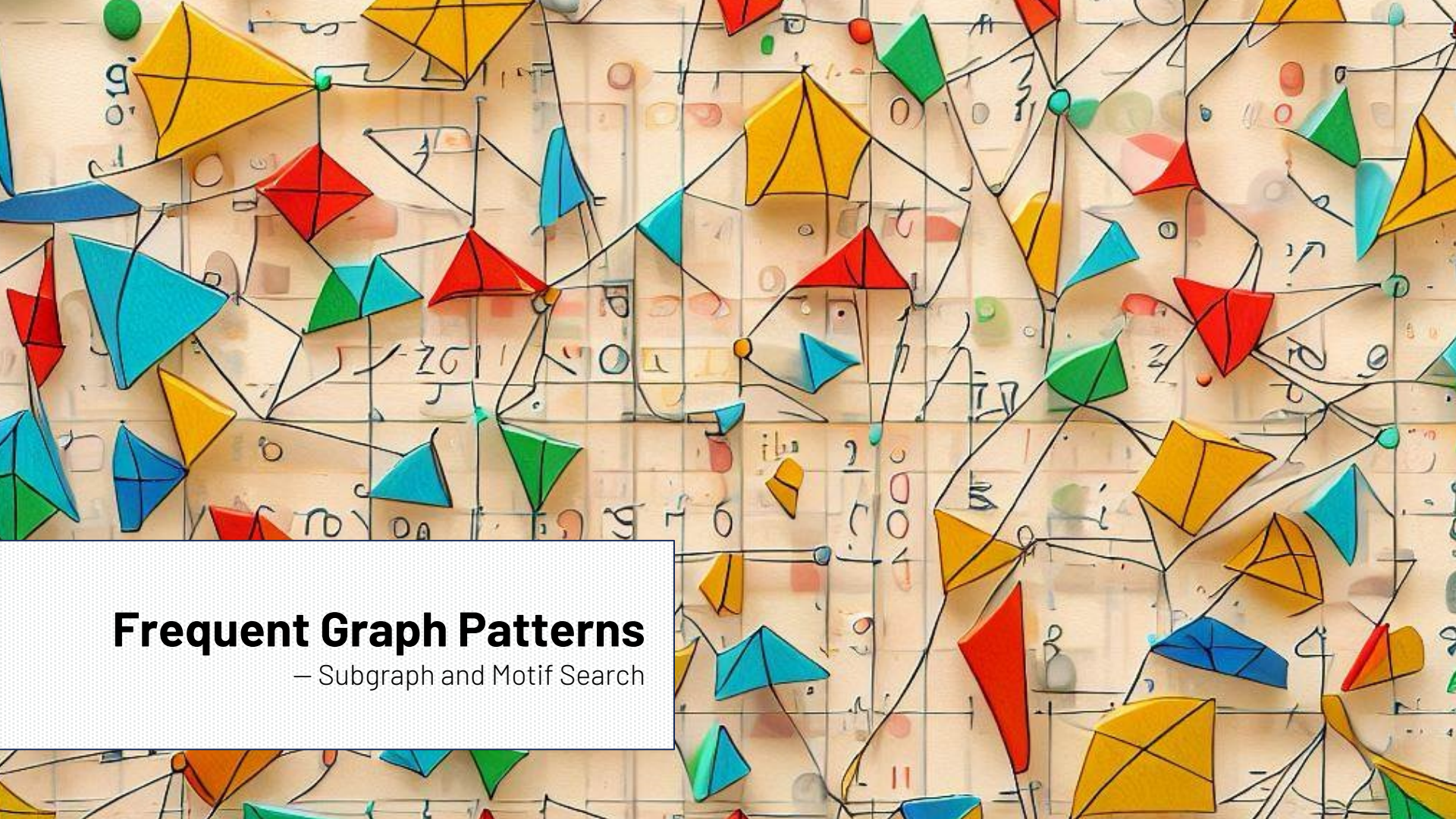
1. **Graph Structure Mining**
   - Graph homomorphism/isomorphism
   - Simulation/Strong Simulation
   - Graphlets/Motifs
   - Triangle Counting
2. **Frequent Graph Pattern Mining**
   - Graph-level subgraph frequency
   - Frequent Subgraph Mining
   - Extracting Shapes

3. **Graph Clustering & Communities**
   - Connected components & Cliques
   - K-core & K-core decomposition
   - Community detection
   - Graph Partitioning & Conductance
   - Girvan Newman algorithm
   - Overlapping Communities
4. **Network Sampling**
   - Scale-down / Back-in-time sampling
   - Induced vs. Incident graph sampling
   - Biased Node Sampling
   - Shape Sampling

Outline

1. **Graph Structure Mining**
   - Graph homomorphism/isomorphism
   - Simulation/Strong Simulation
   - Graphlets/Motifs
   - Triangle Counting
2. **Frequent Graph Pattern Mining**
   - Graph-level subgraph frequency
   - Frequent Subgraph Mining
   - Extracting Shapes

3. **Graph Clustering & Communities**
   - Connected components & Cliques
   - K-core & K-core decomposition
   - Community detection
   - Graph Partitioning & Conductance
   - Girvan Newman algorithm
   - Overlapping Communities
4. **Network Sampling**
   - Scale-down / Back-in-time sampling
   - Induced vs. Incident graph sampling
   - Biased Node Sampling
   - Shape Sampling



Frequent Graph Patterns

— Subgraph and Motif Search

A detour: Association Rules

- **If someone buys diapers and milk, then they are likely to buy beer**
  - Don't be surprised to find beers placed next to diapers!

Supermarket shelf management **– Market-basket model:**

- **Goal:** Identify items that are **bought together** by sufficiently many customers
- **Approach:** Process the sales data collected with barcode scanners to **find dependencies among items**

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

The Market-Basket Model

- A large set of **items**
  - e.g., things sold in a supermarket
- A **large set** of **baskets**
- Each basket is a **small subset of items**
  - e.g., the things one customer buys on one day
- Want to discover **association rules**
  - People who bought {x,y,z} tend to buy {v,w}
    - Amazon!

Input:

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Output:

Rules Discovered:

{Milk} --> {Coke}

{Diaper, Milk} --> {Beer}

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Mining Association Rules

- **Basic Components:**
  1. Frequent Itemsets
  2. Association rule:
     - **Confidence, Support, Interestingness**

- **Algorithms:**
  - Finding Frequent Pairs
  - A-Priori Algorithm

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Frequent Itemsets

- **Simplest question:** Find sets of items that appear together "frequently" in baskets
- ***Support*** for itemset I: Number of baskets containing all items in I
  - (Often expressed as a fraction of the total number of baskets)
- Given a ***support threshold*** s, the sets of items that appear in at least s baskets are called ***frequent itemsets***

Input:

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Support of
{Beer, Bread} = 2/5 = 0.4

A-priori principle:

if {A,B,C} is **frequent** → {A,B} is also **frequent**

if {A,B} is **not frequent** → {A,B,C} **cannot** be frequent

Frequent Graph Pattern Mining

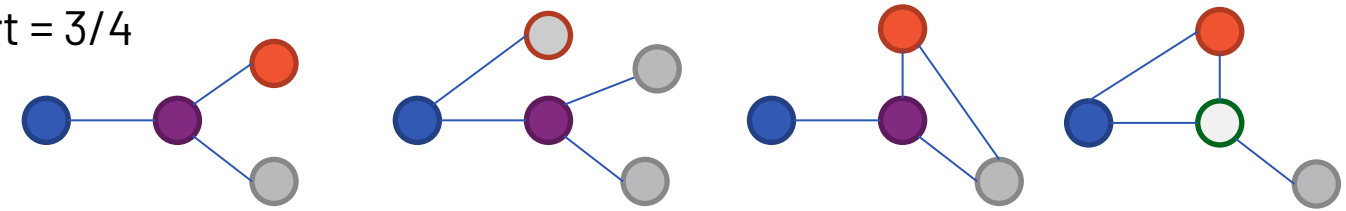
- Frequent subgraphs
  - **A (sub)graph is *frequent* if its support (occurrence frequency) in a dataset is no less than a minimum support threshold**

- Applications of graph pattern mining:
  - Mining biochemical structures
  - Program control flow analysis
  - Mining Social/Web communities
  - Building blocks for graph classification, clustering, compression, comparison

Graph Pattern Mining - Database (Set) of graphs

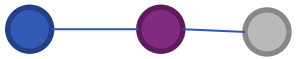
Problem
Given a set of graphs $\{G_1, G_2, \ldots, G_N\}$ find pattern P that appear at least in σ of them

Min support = 3/4

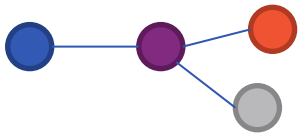


G1 G2 G3 G4

Frequent

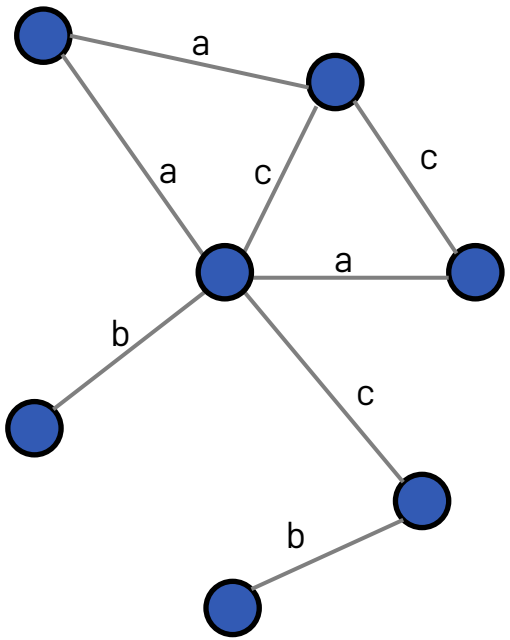


Non-Frequent



Support: frequency of a subgraph appearing in a **set** of graphs

Frequent Subgraph Mining – Single Large Graph



G

Problem
Find all subgraphs of G that appear at least σ times

Suppose $\sigma = 2$, the frequent subgraphs are (only edge labels)

- a, b, c
- a-a, a-c, b-c, c-c
- a-c-a …

Exponential number of patterns!!!

Support over a single large graph

A support measure is **admissible** if for any pattern P and any sub-pattern $Q \sqsubset P$ the **support $\sigma(P)$ is not larger** than support of Q, i.e., $\sigma(Q) \geq \sigma(P)$.
This is also referred **as anti-monotonicity** property.

Problem: The number of occurrences of a pattern in a single large graph is not an admissible support measure

$P1 \sqsubset P2$, but $1 = \sigma(P1) < \sigma(P2) = 2$

How can we solve this problem?

Support over a single large graph

A number of admissible support measures have been proposed:

1. **Maximum Independent Set support** (MIS)
   - Based on maximum number of **non-overlapping matches**
2. **Harmful overlap support** (HO)
   - Based on the number of patterns for **which no (multi-node) subgraph** is identical
3. **Minimum Image-based support** (MNI)
   - Based on the **number of times a node in the pattern** is mapped into a distinct node in the graph

Maximum independent set support (MIS)

- **Idea:** Count how many times a pattern is mapped to a **non-overlapping subgraph**

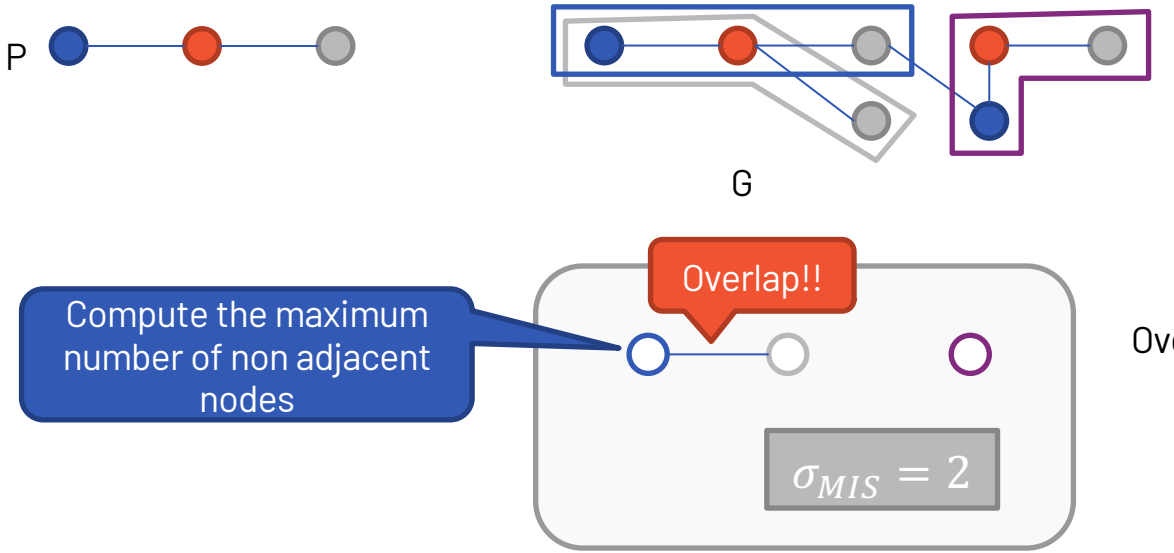
MIS computation

1. Construct an **overlap graph** $\mathrm{G_O}: \langle \mathrm{V_O}, \mathrm{E_O} \rangle$, in which the set of nodes $\mathrm{V_O}$ is the set of matches of a pattern P into a graph G and
   $E_O = \{(f_1, f_2) | f_1, f_2 \in V_O \wedge f_1 \neq f_2 \wedge f_1 \sqcap f_2 \neq \emptyset\}$,
   i.e., $\mathrm{E_O}$ has an an edge among each pair of overlapping matches.

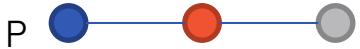
2. σ_{MIS} = size of the **maximum independent set** of nodes in the overlap graph.

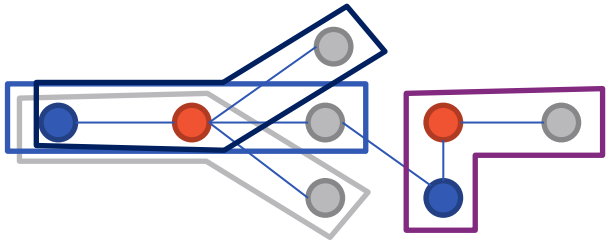
- An **independent set** of nodes in a graph is a subset of non-adjacent nodes of the graph, i.e. $\mathrm{G}: \langle V, E \rangle, I \subseteq V$ s.t. $\forall (u, v) \in I, (u, v) \notin E$
- The biggest possible independent set is called the **maximum independent set**

Maximum independent set support (MIS) : Example

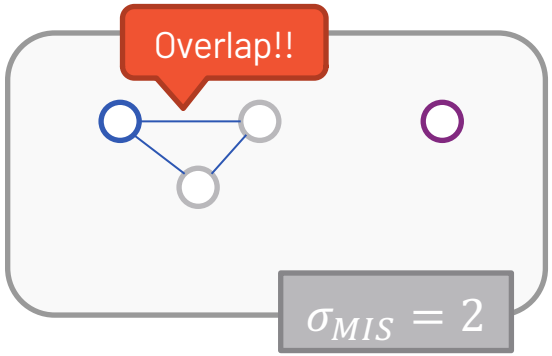


Maximum independent set support (MIS) : Example 2





Compute the maximum number of non adjacent nodes

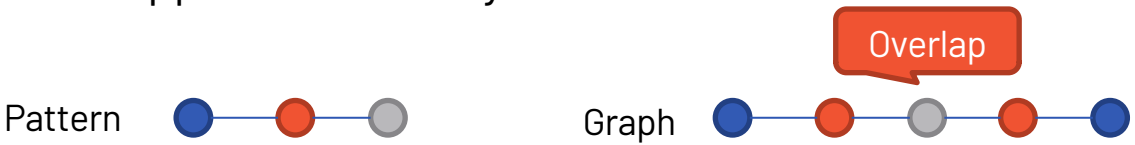


Overlap graph: G_O

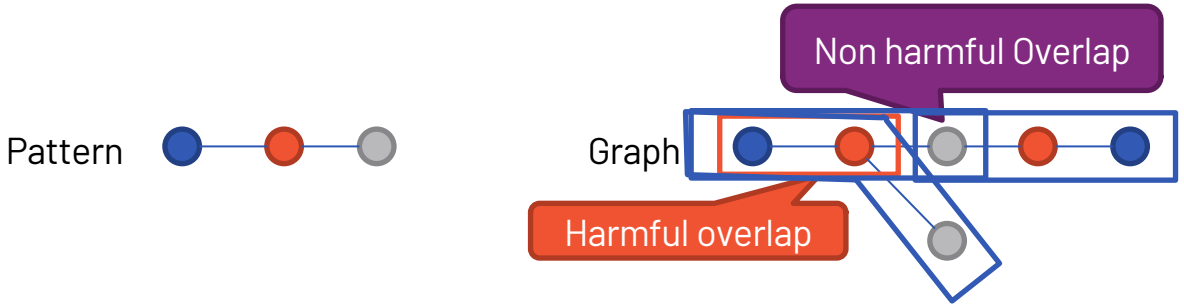
Finding a maximum independent set is NP-hard → Very High computational cost

Harmful overlap support (HO)

- MIS support can be very restrictive and considers overlap among *single nodes*



- Harmful overlap support σ_{HO} considers as *harmful* those *subgraphs* that share a ***common (multi-node) subgraph***



Support is computed like in MIS, the difference is how we compute the overlap graph

Fiedler, M. and Borgelt, C., 2007. Subgraph support in a single large graph. *ICDMW 2007.*

Minimum Image-based support (MNI) : Example

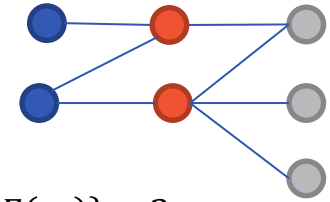
Simpler support based on the minimum number of times a node of the pattern is mapped to the graph

Let $f_1, \dots, f_m$ be the set of isomorphisms of a pattern P: $\langle V_P, E_P, \ell_P \rangle$ in a graph G. Let $F(v) = |\{f_1(v), \dots, f_m(v)\}|$ be the number of distinct mappings of a node $v \in V_P$ to a node in G by functions $f_1, \dots, f_m$. The **Minimum Image-based support (MNI)** $\sigma_{MNI}(P)$ of P in G is $\sigma_{MNI}(P) = \min\{F(v), v \in V_P\}$

Pattern

$F(v_1) = 2$
$F(v_2) = 2$
$F(v_3) = 3$

Graph



$\sigma_{MNI}(P) = \min\{F(v_1), F(v_2), F(v_3)\} = 2$

Chen, C., Yan, X., Zhu, F. and Han, J. gapprox: Mining frequent approximate patterns from a massive network. ICDM, 2007.

Properties of the support measures

- MIS, HO, and MNI are anti-monotone (proof omitted)
- Computation time:
  - MIS and HO are **NP-**hard to compute (*there is no algorithm to solve them efficiently*)
  - MNI can be computed in polynomial time!
- What is the relationship among the support sets?
  - MIS support set is a subset of HO support set, which is a subset of MNI support set
  - $\sigma_{MIS}(P) \leq \sigma_{HO}(P) \leq \sigma_{MNI}(P)$

Mohammed Elseidy, Ehab Abdelhamid, Spiros Skiadopoulos, and Panos Kalnis. 2014. GraMi: frequent subgraph and pattern mining in a single large graph. Proc. VLDB Endow. 7, 7 (March 2014), 517–528. **https://github.com/ehab-abdelhamid/GraMi**

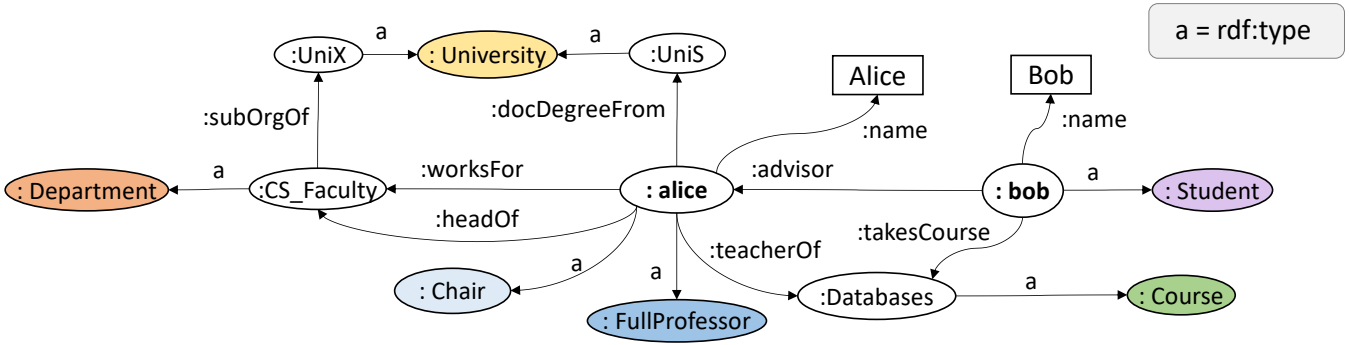
Mining Patterns from a Query log

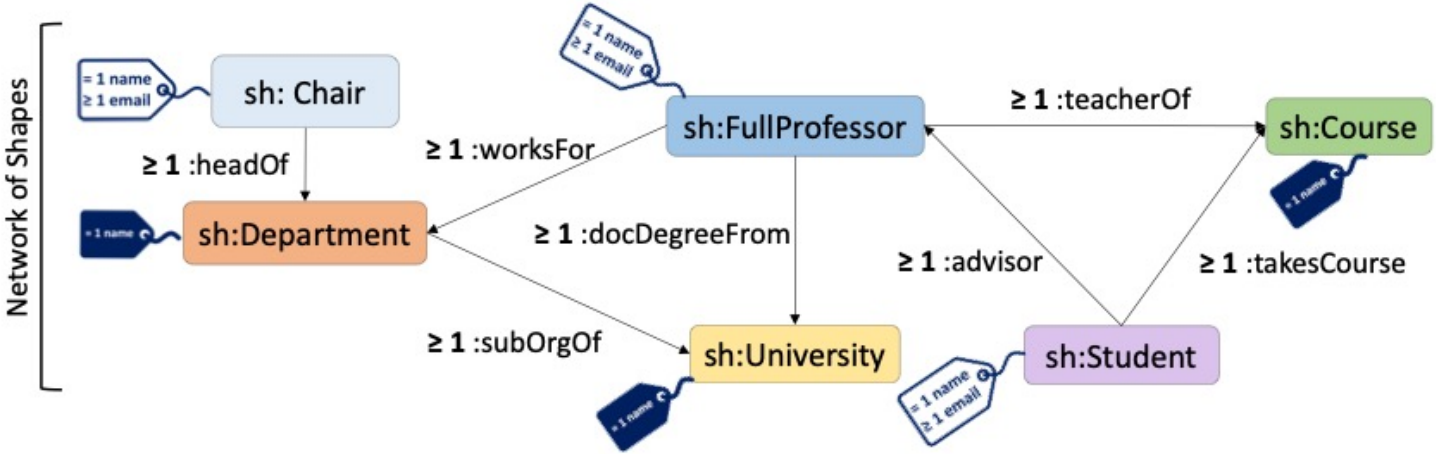
You have access to the log of all SPARQL queries submitted to a large KG DBMS.
(e.g., http://lsq.aksw.org/)

Suppose you want to apply a graph pattern mining approach to this data.
Define how you would approach it. What is a pattern?
What can a pattern tell you? How can you use this information?

https://bit.ly/ACM24-E04

Mining Graph Shapes: a.k.a. structural summarization



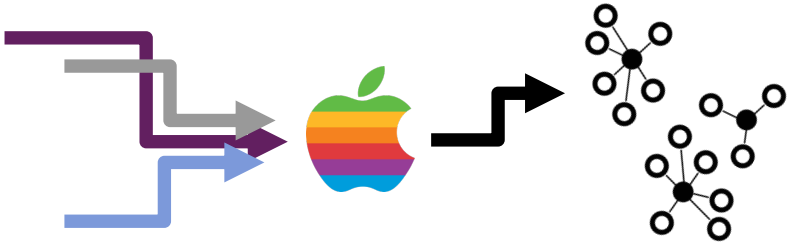


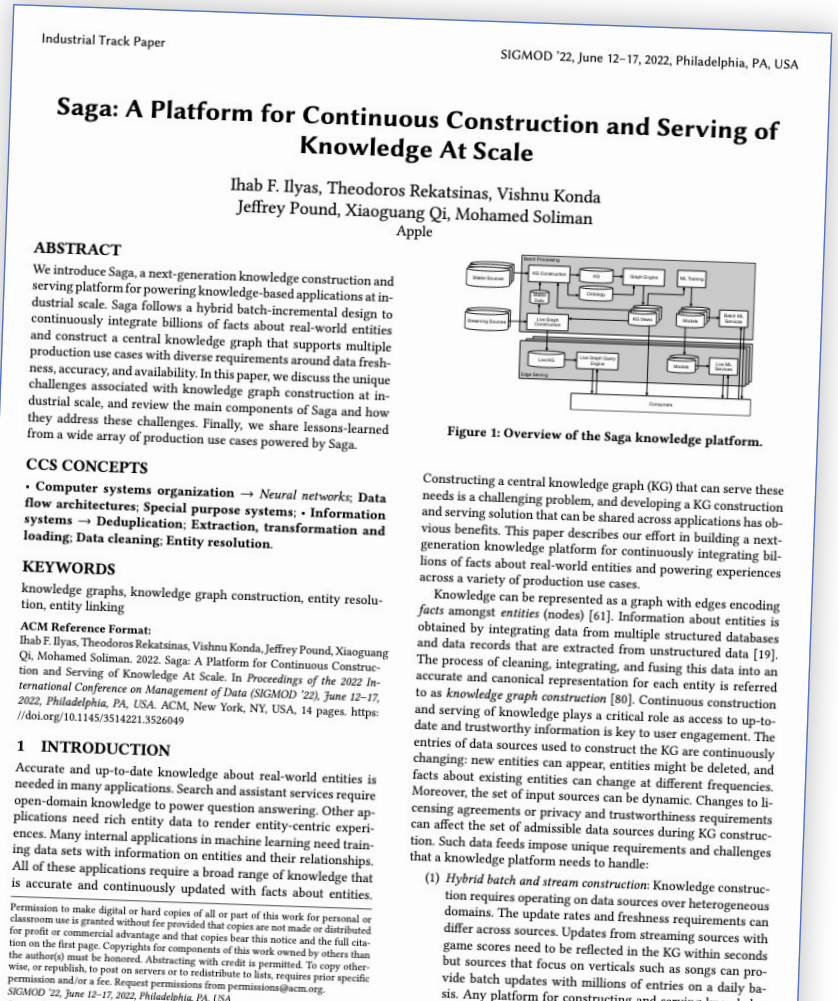
Enterprise Knowledge Graphs

A data model flexible and expressive to integrate large heterogenous datasets

> *The entries of data sources used to construct the KG* ***are continuously changing****...*
>
> ***...onboarding of new data sources*** *is important to ensure consistent growth of the KG.*

No need to define a schema before inserting data....
No need for a unified schema....





Industrial Track Paper SIGMOD '22, June 12–17, 2022, Philadelphia, PA, USA

Saga: A Platform for Continuous Construction and Serving of Knowledge At Scale

Ihab F. Ilyas, Theodoros Rekatsinas, Vishnu Konda
Jeffrey Pound, Xiaoguang Qi, Mohamed Soliman
Apple

ABSTRACT

We introduce Saga, a next-generation knowledge construction and serving platform for powering knowledge-based applications at industrial scale. Saga follows a hybrid batch-incremental design to continuously integrate billions of facts about real-world entities and construct a central knowledge graph that supports multiple production use cases with diverse requirements around data freshness, accuracy, and availability. In this paper, we discuss the unique challenges associated with knowledge graph construction at industrial scale, and review the main components of Saga and how they address these challenges. Finally, we share lessons-learned from a wide array of production use cases powered by Saga.

CCS CONCEPTS

• **Computer systems organization** → *Neural networks*; **Data flow architectures**; **Special purpose systems**; • **Information systems** → **Deduplication**; **Extraction, transformation and loading**; **Data cleaning**; **Entity resolution**.

KEYWORDS

knowledge graphs, knowledge graph construction, entity resolution, entity linking

ACM Reference Format:
Ihab F. Ilyas, Theodoros Rekatsinas, Vishnu Konda, Jeffrey Pound, Xiaoguang Qi, Mohamed Soliman. 2022. Saga: A Platform for Continuous Construction and Serving of Knowledge At Scale. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22), June 12–17, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3514221.3526049

1 INTRODUCTION

Accurate and up-to-date knowledge about real-world entities is needed in many applications. Search and assistant services require open-domain knowledge to power question answering. Other applications need rich entity data to render entity-centric experiences. Many internal applications in machine learning need training data sets with information on entities and their relationships. All of these applications require a broad range of knowledge that is accurate and continuously updated with facts about entities.

SIGMOD '22, June 12–17, 2022, Philadelphia, PA, USA

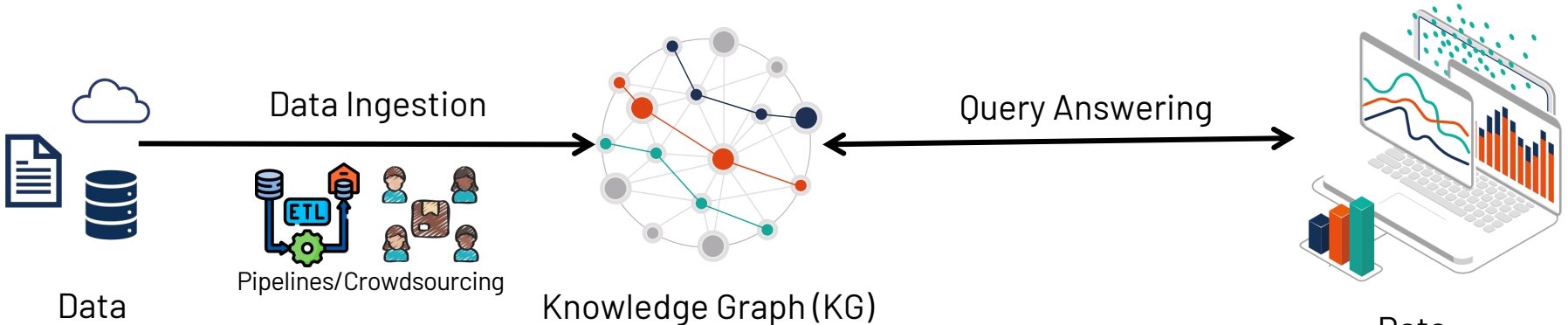
Figure 1: Overview of the Saga knowledge platform.

Constructing a central knowledge graph (KG) that can serve these needs is a challenging problem, and developing a KG construction and serving solution that can be shared across applications has obvious benefits. This paper describes our effort in building a next-generation knowledge platform for continuously integrating billions of facts about real-world entities and powering experiences across a variety of production use cases.

Knowledge can be represented as a graph with edges encoding *facts* amongst *entities* (nodes) [61]. Information about entities is obtained by integrating data from multiple structured databases and data records that are extracted from unstructured data [19]. The process of cleaning, integrating, and fusing this data into an accurate and canonical representation for each entity is referred to as *knowledge graph construction* [80]. Continuous construction and serving of knowledge plays a critical role as access to up-to-date and trustworthy information is key to user engagement. The entries of data sources used to construct the KG are continuously changing: new entities can appear, entities might be deleted, and facts about existing entities can change at different frequencies. Moreover, the set of input sources can be dynamic. Changes to licensing agreements or privacy and trustworthiness requirements can affect the set of admissible data sources during KG construction. Such data feeds impose unique requirements and challenges that a knowledge platform needs to handle:

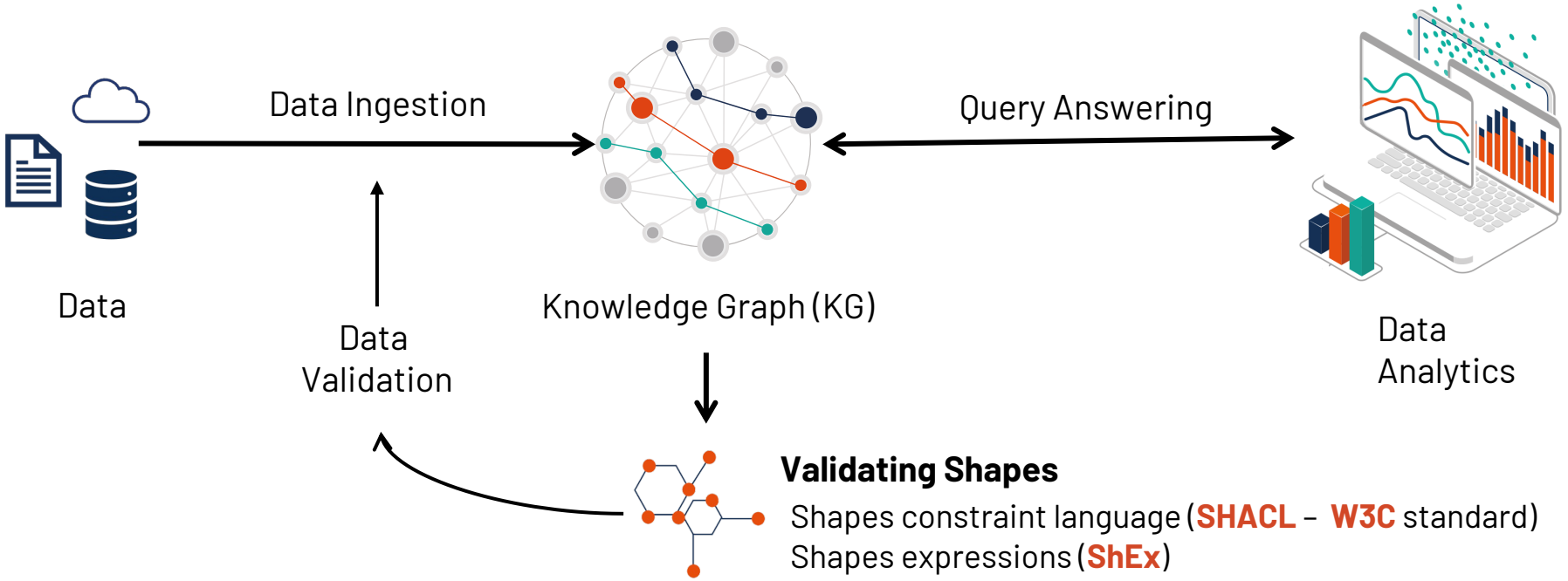
(1) *Hybrid batch and stream construction:* Knowledge construction requires operating on data sources over heterogeneous domains. The update rates and freshness requirements can differ across sources. Updates from streaming sources with game scores need to be reflected in the KG within seconds but sources that focus on verticals such as songs can provide batch updates with millions of entries on a daily basis. Any platform for constructing and

Data quality in KGs



- What about the **quality** of the **data** in the KG?
- How to **validate** the **data** in the KG?
- How to **avoid insertion** of **erroneous** data in the **data ingestion** phase?

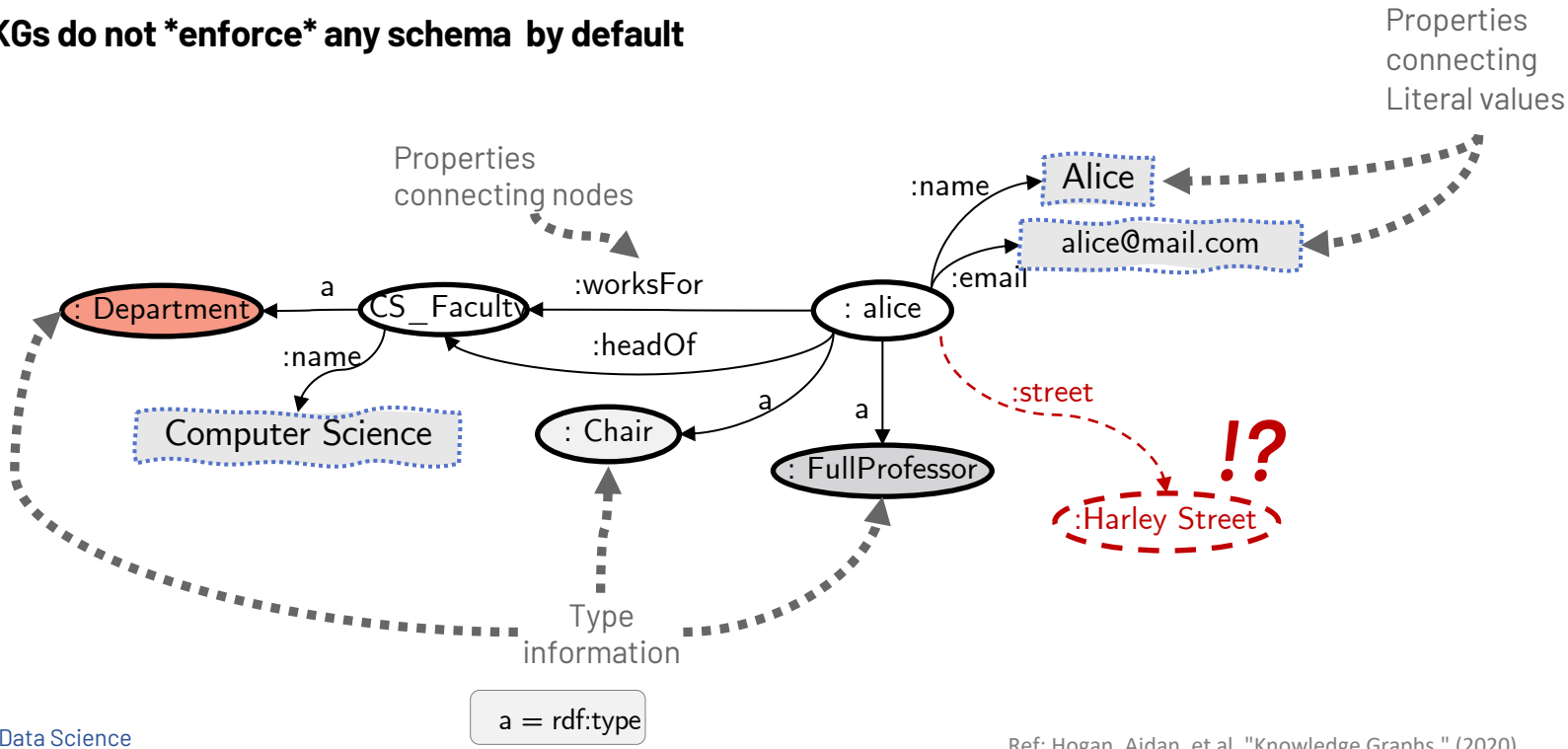
Data quality in KGs: validating shapes



? **How to define Validating Shapes?**

Knowledge Graphs: RDF with no schema

- **In an RDF KG data values, node connections, and schema information co-exist together**
- **RDF KGs do not *enforce* any schema by default**



Ref: Hogan, Aidan, et al. "Knowledge Graphs." (2020).

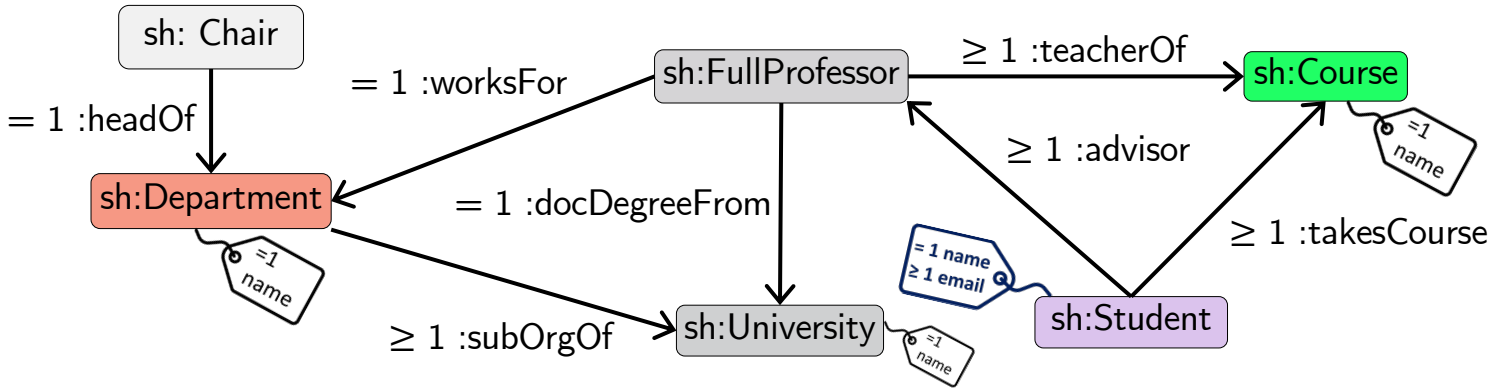
Validating Schemas

- Validating schemas are a "separate" type of RDF Graph **which describe a form of ER schema** for the data within a KG
- They describe **how classes can (or cannot) be connected to other classes or data values**, by which type of relationships, and their allowed cardinality constraints

① **Identify Classes**

② **Identify Properties with Cardinality**

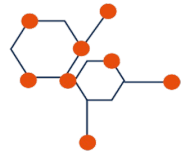
- Literal
- Non-Literal



Defining Validating Shapes?

1 The most **reliable** way is to define them **manually.**

Yet, **KGs are very large in size (up to 100 millions of entities in WikiData and 100s of thousands of classes),**
thus, it requires too much manual effort



Validating Shapes
SHACL/ShEx

KGs

Automatic Extraction of Shapes

1 The most **reliable** way is to define them **manually.**

Yet, **KGs are very large in size (up to 100 millions of entities in WikiData and 100 of thousands of classes)**, **thus, it requires too much manual effort**

2 **Alternatively, we can mine them from the data**

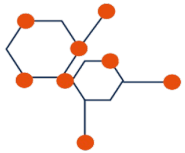
Existing shapes extraction approaches **derive** shape constraints from **data**, by **trusting** the content of the KG.

Crowd Sourcing

★★☆ **LOW QUALITY** OF THE PRODUCED SHAPES

KGs

Extractors



Validating Shapes

SHACL/ShEx

Judging Quality of Extracted Shapes

QUALITY OF THE OUTPUT SHAPES

%?

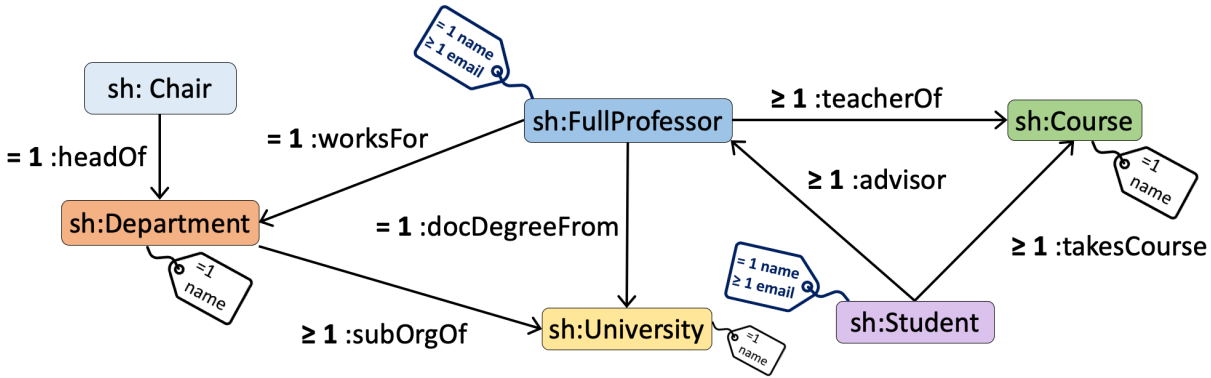
concept of **support** and **confidence** (Frequent Pattern Mining)

- **Support**: **number of entities of a specific type/class**,

  support of **property shape** is defined as the **number** of entities conforming to its constraints.

- **Confidence**: the **ratio** between ***how many entities conform to a specific constraint***

  and the ***total number of entities for the target class of the node shape***.



$Conf(takesCourse) = {}^{5}/_{10}$

$|Entities_{takesCourse}| = 5$

$|Entities_{Student}| = 10$

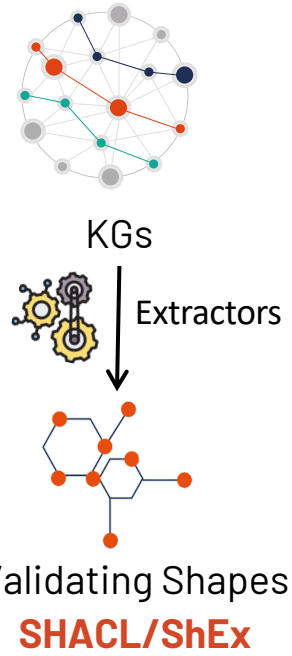
Validating Shape Extraction Problem

Given a Knowledge Graph (stored in a file or behind a SPARQL endpoint)

a) Extract all the Node & Property shapes, and

b) Annotate each Node & Property shape with Support & Confidence annotation

c) Extract only Shapes with High Support/Confidence



Kashif Rabbani, Matteo Lissandrini, and Katja Hose. 2023. Extraction of Validating Shapes from Very Large Knowledge Graphs. VLDB 2023.
https://doi.org/10.14778/3579075.3579078
See also: https://relweb.cs.aau.dk/qse/

Outline

1. **Graph Structure Mining**
   - Graph homomorphism/isomorphism
   - Simulation/Strong Simulation
   - Graphlets/Motifs
   - Triangle Counting
2. **Frequent Graph Patterns**
   - Graph-level subgraph frequency
   - Frequent Subgraph Mining
   - Extracting Shapes

3. **Graph Clustering & Communities**
   - Connected components & Cliques
   - K-core & K-core decomposition
   - Community detection
   - Graph Partitioning & Conductance
   - Girvan Newman algorithm
   - Overlapping Communities
4. **Network Sampling**
   - Scale-down / Back-in-time sampling
   - Induced vs. Incident graph sampling
   - Biased Node Sampling
   - Shape Sampling

Further References

Davis Shurbert, An Introduction to Graph Homomorphisms
http://buzzard.ups.edu/courses/2013spring/projects/davis-homomorphism-ups-434-2013.pdf

Andreas Schmidt, Iztok Savnik, Conference on Advances in Databases, Knowledge, and Data Applications, Overview of Regular Path Queries in Graphs
https://www.iaria.org/conferences2015/filesDBKDA15/graphsm_overview_of_regular_path_queries_in_graphs.pdf

Jure Leskovec, CS224W: Machine Learning with Graphs | 2019 |
Lecture 3-Motifs and Structural roles in networks
http://snap.stanford.edu/class/cs224w-2019/slides/03-motifs.pdf

Davide Mottin and Emmanuel Müller, Graph Exploration:
Let me Show what is relevant in your graph, KDD tutorial
https://mott.in/slides/KDD2018-Tutorial-Compressed.pdf

Kolaczyk, E.D., Statistical Analysis of Network Data, Chapter 5: Sampling and Estimation in Network Graphs.
https://link.springer.com/chapter/10.1007/978-0-387-88146-1_5

Jure Leskovec, Christos Faloutsos, Sampling from Large Graphs
https://dl.acm.org/doi/pdf/10.1145/1150402.1150479