QS STARS RATED FOR EXCELLENCE 2012

Fpt University

TRƯỜNG ĐẠI HỌC FPT

Capstone Project Defense

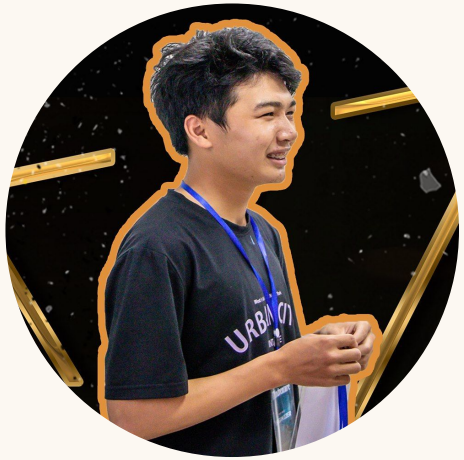
An Improvement of Cluster-GCN with constraints

Presenter: AIP490_G7

Supervisor: Assoc. Prof. Phan Duy Hung

TEAM MEMBERS

Duong Thuy Trang
HE150573
trangdthe150573@fpt.edu.vn



Nguyen Thanh Tung
HE150163
tungnthe150163@fpt.edu.vn

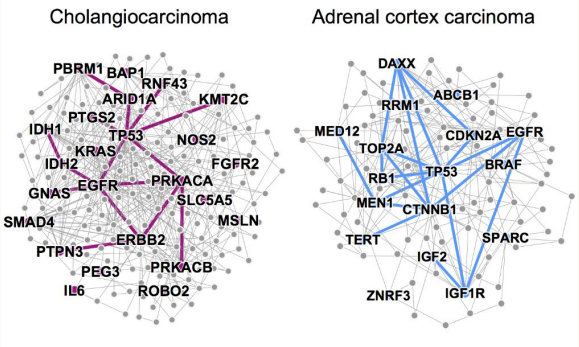
Nguyen Bao Phuoc
HE153036
phuocnbhe153036@fpt.edu.vn

Table of contents

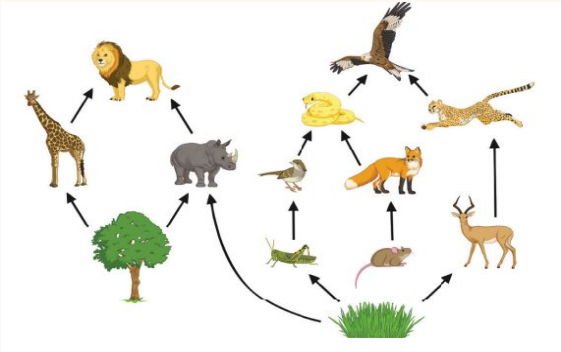
01 Introduction

- Graph are used to describe and analyze entities with interactions/ relations
- The explosive growth in volume of data poses challenges for deep learning algorithm, esp **GCN**
- Training large graphs typically requires expensive hardware that students can’t afford

- Disease pathway graphs



- Food webs



MOTIVATION

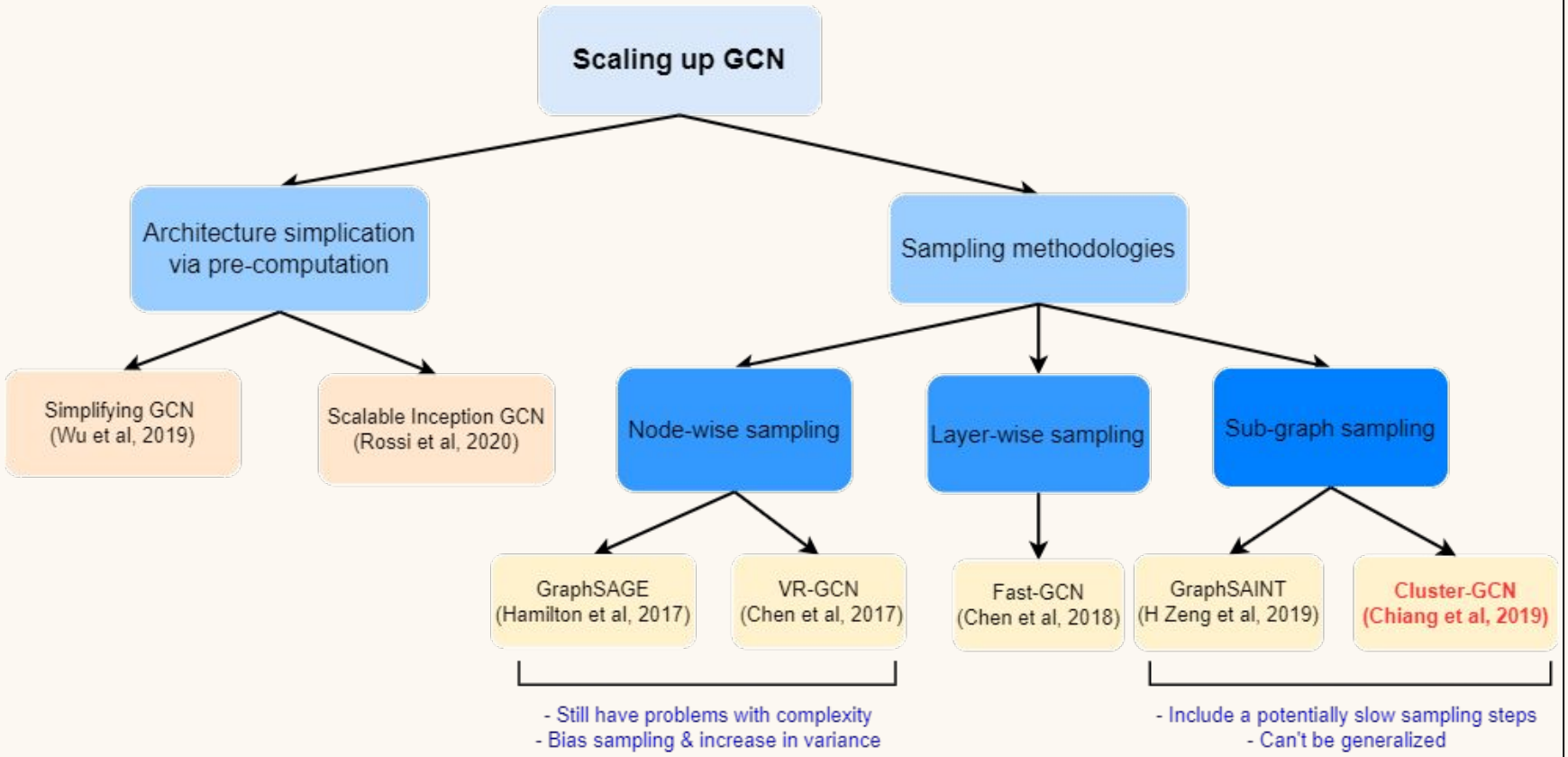
GCN is a good architecture for learning large graphs, but it suffers from **slow convergence** and **out-of memory** issue

→ We aim to study strategies to scale GCN

02 Related works

- Previous approach to scale GCNs
- Community detection and METIS algorithm.

SCALABILITY OF GCNs



COMMUNITY DETECTION

METIS algorithm was proposed in paper: "*A fast and high quality multilevel scheme for partitioning irregular graphs*." by Karypis et al. in 1998

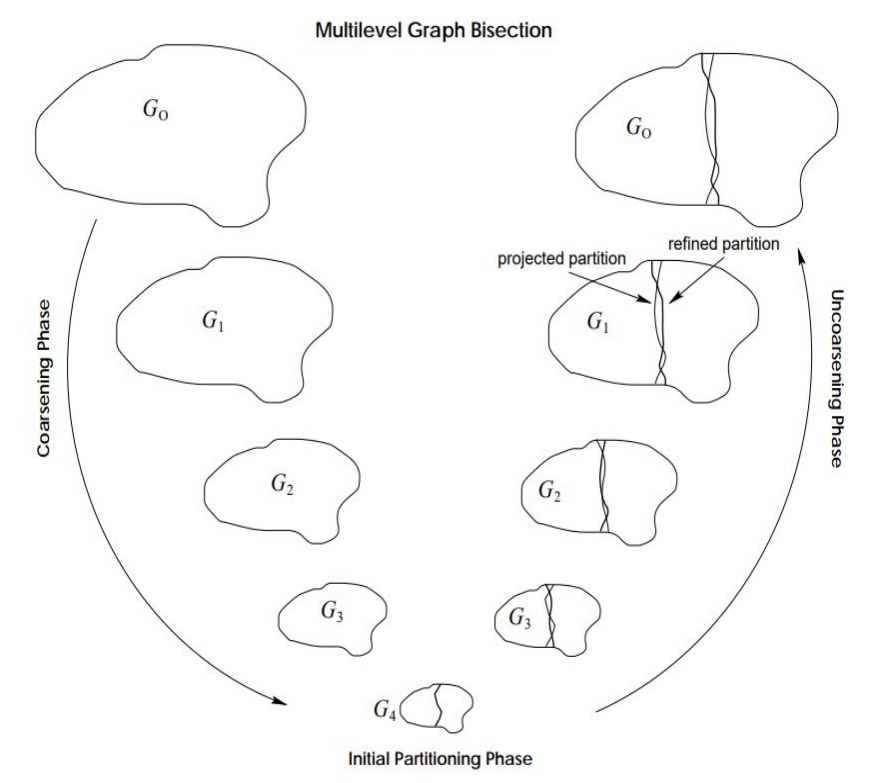


Figure: An illustration of METIS algorithm

COMMUNITY DETECTION

In 2004, NEWMAN et al. proposes *modularity* as a metric to evaluate clustering results

⇒ **Modularity-based algorithms** (e.g., Louvain, Leiden) is proven to be better than METIS[2,3,4]

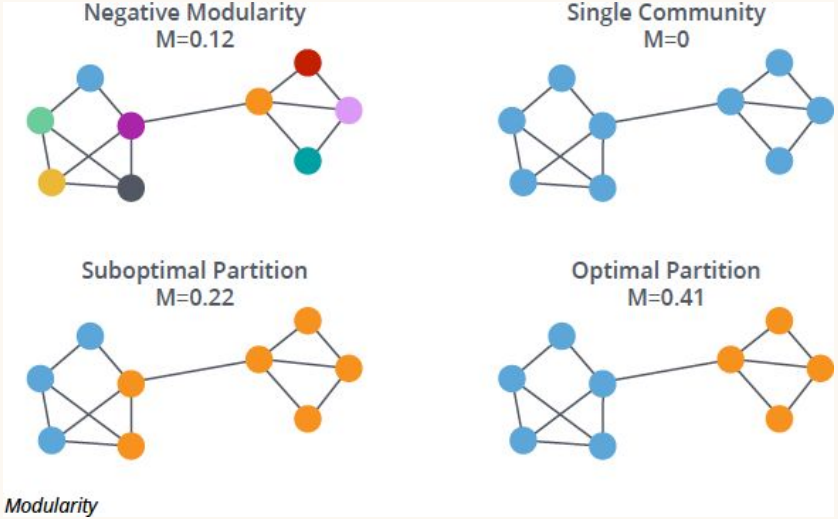


Figure: An illustration of modularity

[1] NEWMAN, Mark EJ; GIRVAN, Michelle. Finding and evaluating community structure in networks. Physical review E, 2004, 69.2: 026113.
[2] SHIOKAWA, Hiroaki; ONIZUKA, Makoto. Scalable Graph Clustering and Its Applications. 2018.
[3]LIU, Yike; SHAH, Neil; KOUTRA, Danai. An empirical comparison of the summarization power of graph clustering methods. arXiv preprint arXiv:1511.06820, 2015.
[4]XU, Hongteng; LUO, Dixin; CARIN, Lawrence. Scalable gromov-wasserstein learning for graph partitioning and matching. Advances in neural information processing systems, 2019, 32.

OBJECTIVE AND CONTRIBUTION

The goal of this thesis is to **improve Cluster-GCN sampling** phase. Our contributions are as follow:

- Testing **Leiden algorithm** in graph-wise sampling phase
- Suggestions in **adding constraints** to Leiden to improve efficiency of Cluster-GCN

03 Data exploration

- Dataset introduction
- Edge feature learning: aggregate edge features into node features

DATA INTRODUCTION

OGBN-PROTEINS DATASET INTRODUCTION

- Dataset designed for node property prediction
- Include a protein-protein association networks, collected from 8 species
- **Challenge**: multi-label protein function prediction, 112 labels in total

#Graphs	#Node	#Edge	#Labels	#Features
1	132534	39561252	112	8

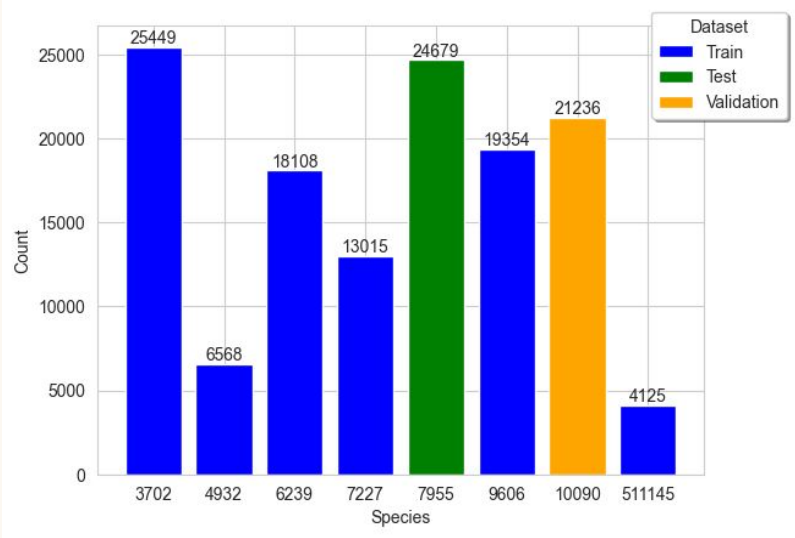


Figure: Distribution of node species among train, validation and test set

EDGE FEATURE LEARNING

Instead of using the species ID, the model constructs a set of node features by **aggregating edge features** of each node's entire neighbors

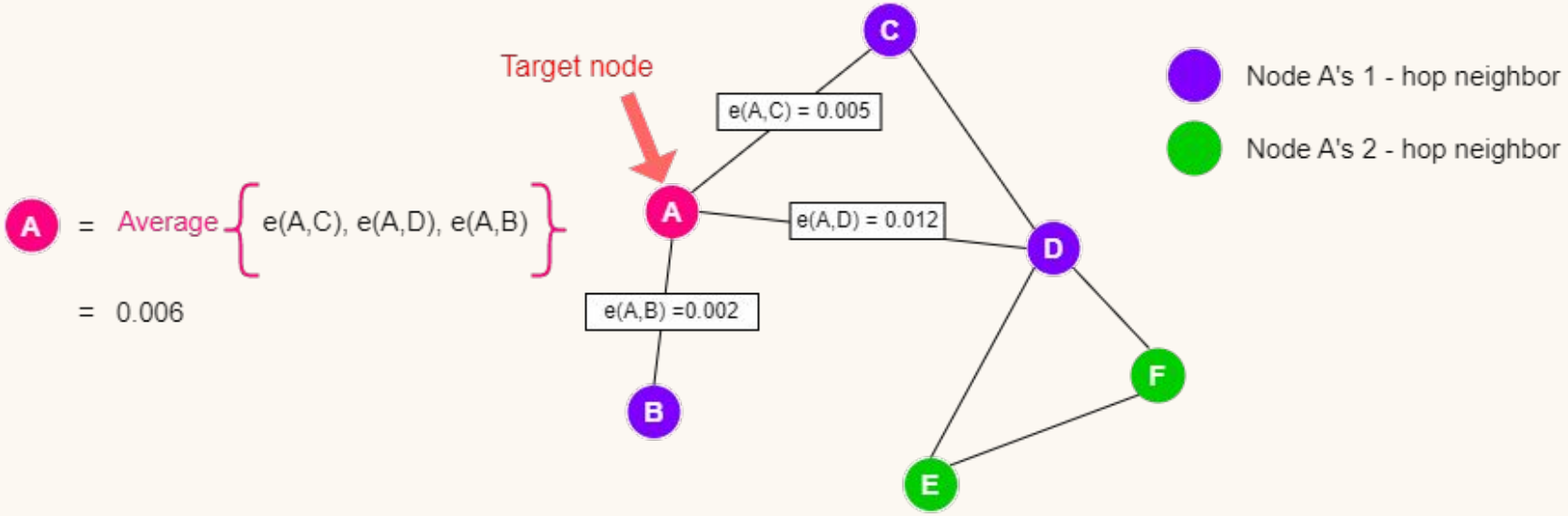
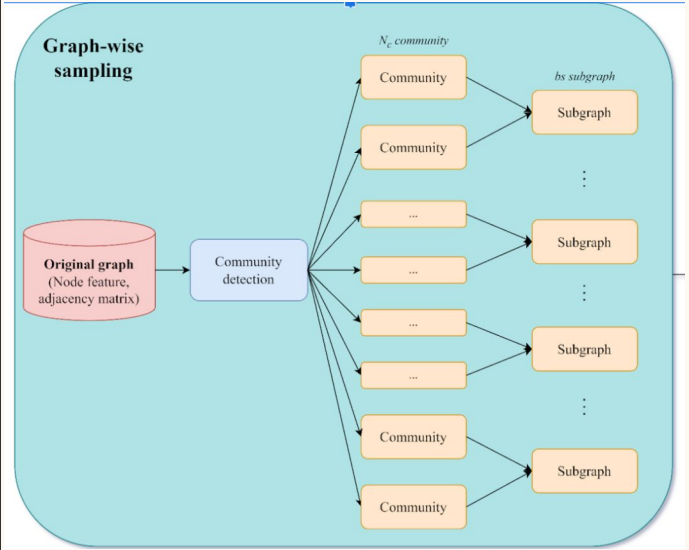


Figure: An illustration of edge feature learning

04 Methodology

- Cluster-GCN
- Leiden algorithm
- Constraints to improve Leiden algorithm

CLUSTER-GCN



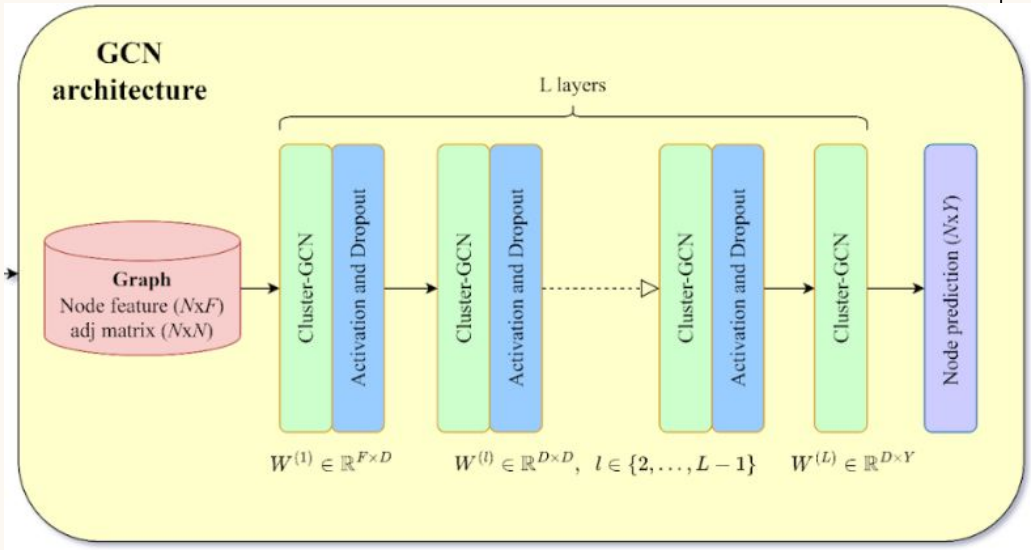


Figure: Baseline architecture.

STOCHASTIC MULTIPLE PARTITIONS

- Proposed to tackle **skewed label distribution**

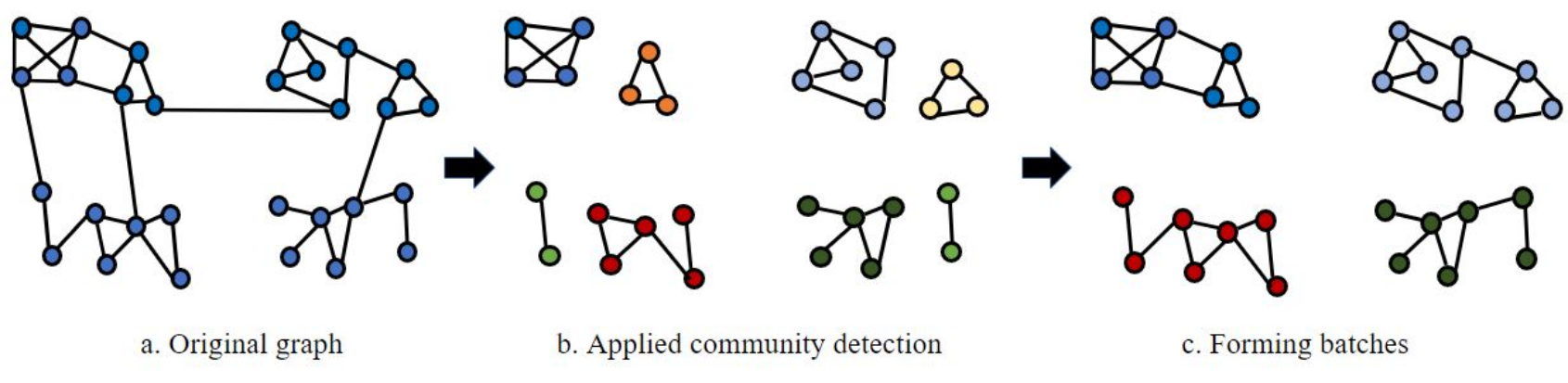


Figure: An illustration of stochastic multiple partitions

LEIDEN ALGORITHM

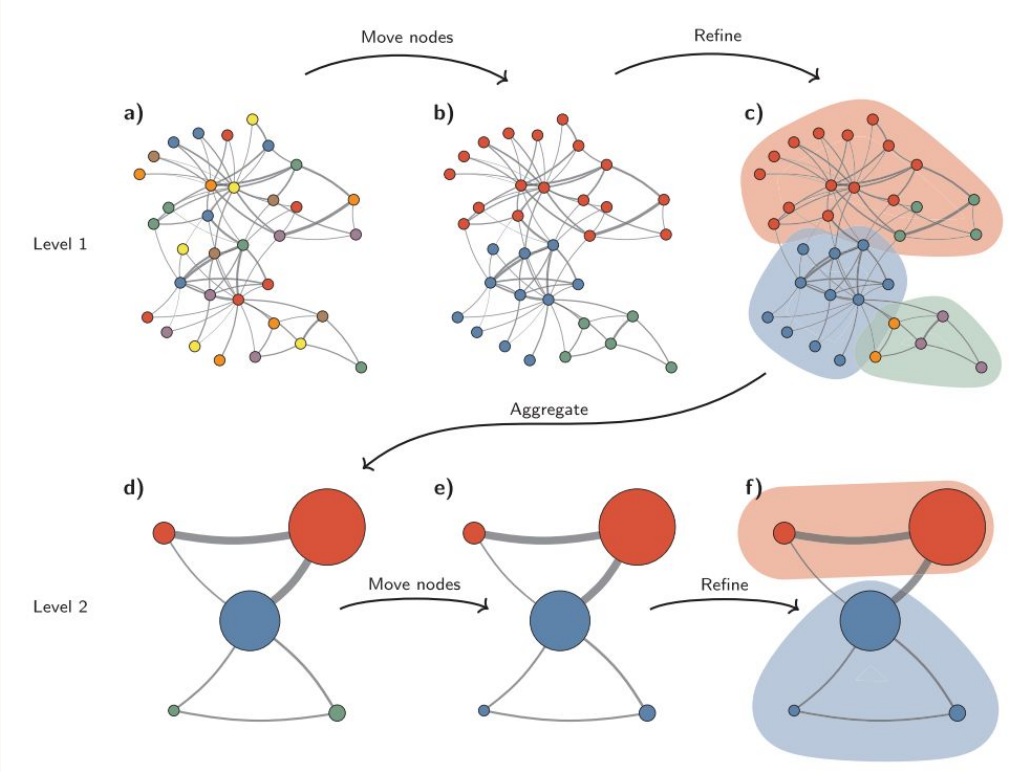


Figure: Illustration of Leiden algorithm

LEIDEN ALGORITHM

⇒ The objective of Leiden algorithm is to **maximizing the modularity (Q):**

$$Q = \frac{1}{2m} \sum_{i,j} \left(\mathbf{A}_{i,j} - \gamma \frac{k_i k_j}{2m} \right) \delta\left(c_i, c_j\right)$$

$$= \frac{1}{2m} \sum_{c=1}^{\mathbf{C}} \left(\mathbf{e}_c - \gamma \frac{K_c^2}{2m} \right)$$

COMMUNITY SIZE CONSTRAINTS

Limitation of Leiden algorithm: resolution limit[1]

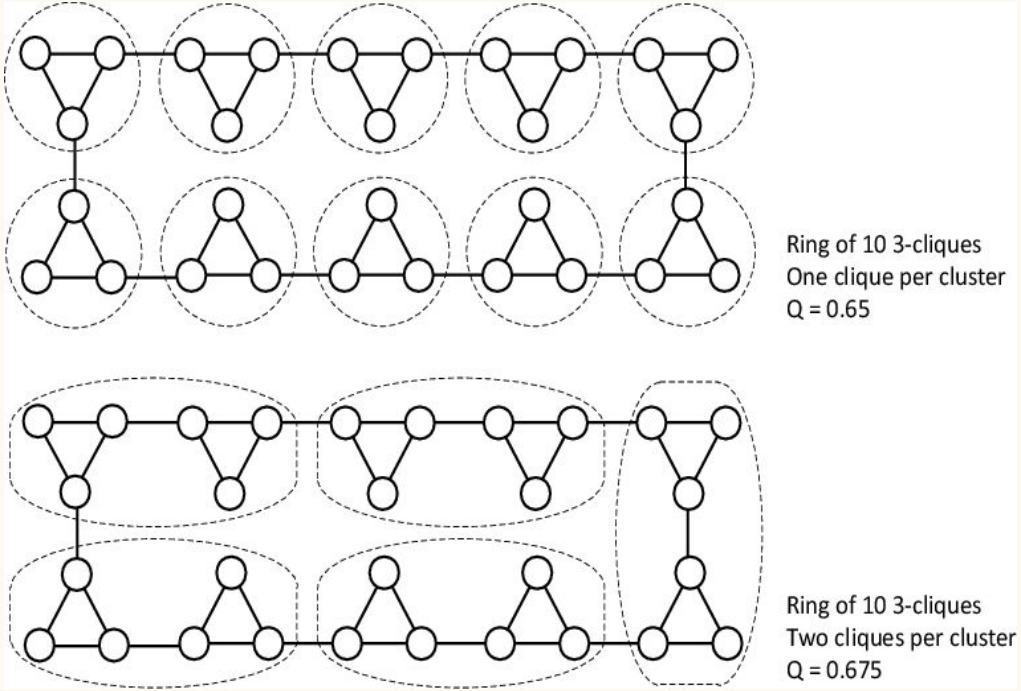


Figure: Resolution limit of modularity

[1]FORTUNATO, Santo; BARTHELEMY, Marc. Resolution limit in community detection. *Proceedings of the national academy of sciences*, 2007, 104.1: 36-41.

COMMUNITY SIZE CONSTRAINTS

Limitation of Leiden algorithm: not guarantee of approximately equal size community

⇒ Proposes constraints: ct_{min} and ct_{max}

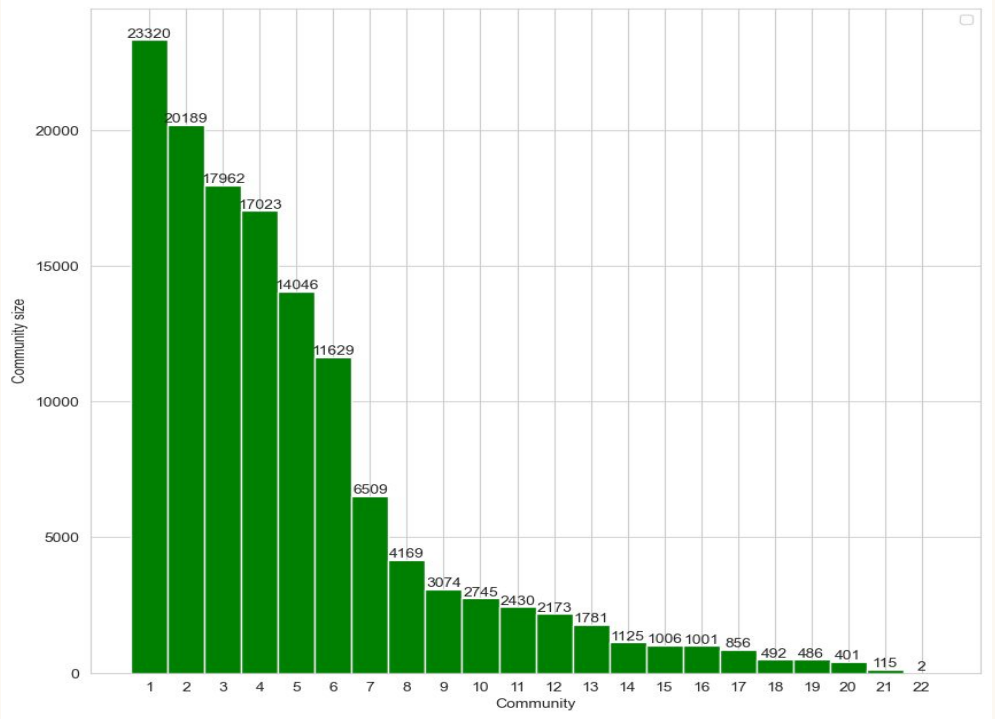


Figure: Size of subgraphs extracted by Leiden algorithm

COMMUNITY SIZE CONSTRAINTS

Edge ratio:

$$ER(C_i, C_j) = \frac{E(C_i, C_j)}{|C_i + C_j|} + eps$$

Probability of merging 2 communities:

$$Pr(C_i = C_j) = \frac{ER(C_i, C_j)}{\sum_{C_k \in (\mathbf{C} - C_i)} ER(C_i, C_k)}$$

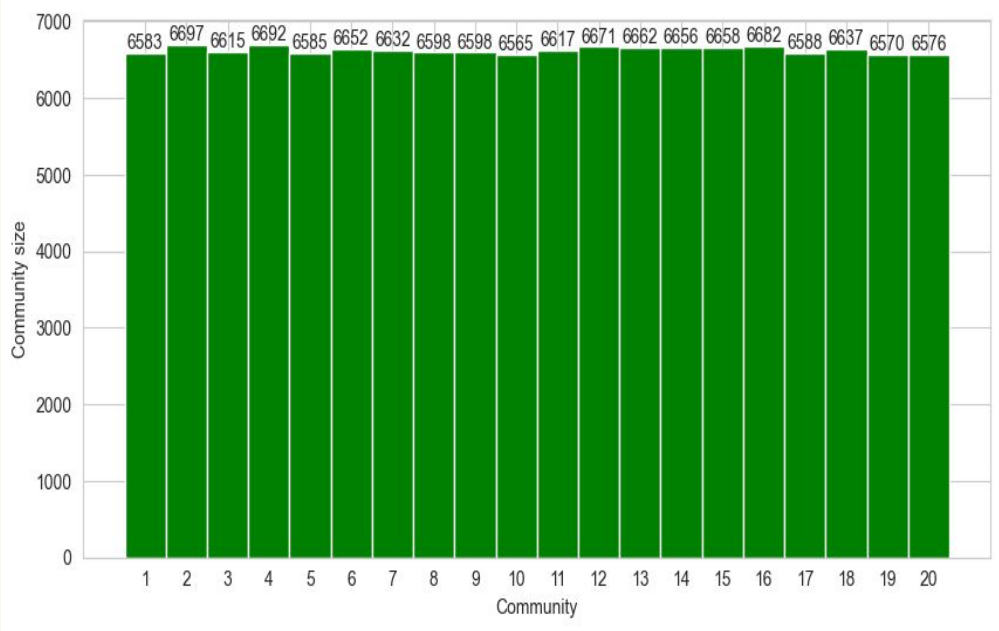


Figure: Size of subgraphs after using maximum/minimum community size constraint.

COMMUNITY SIZE CONSTRAINTS

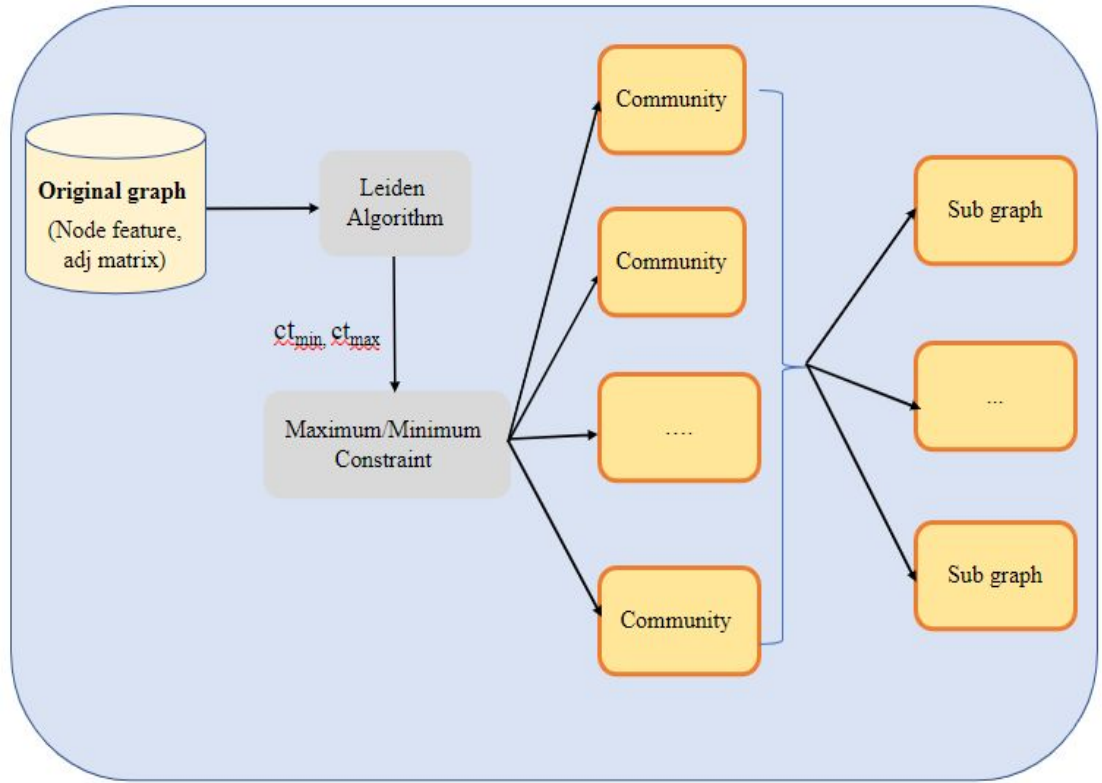


Figure:Leiden algorithm with community size constraints.

OVERLAPPING COMMUNITY CONSTRAINTS

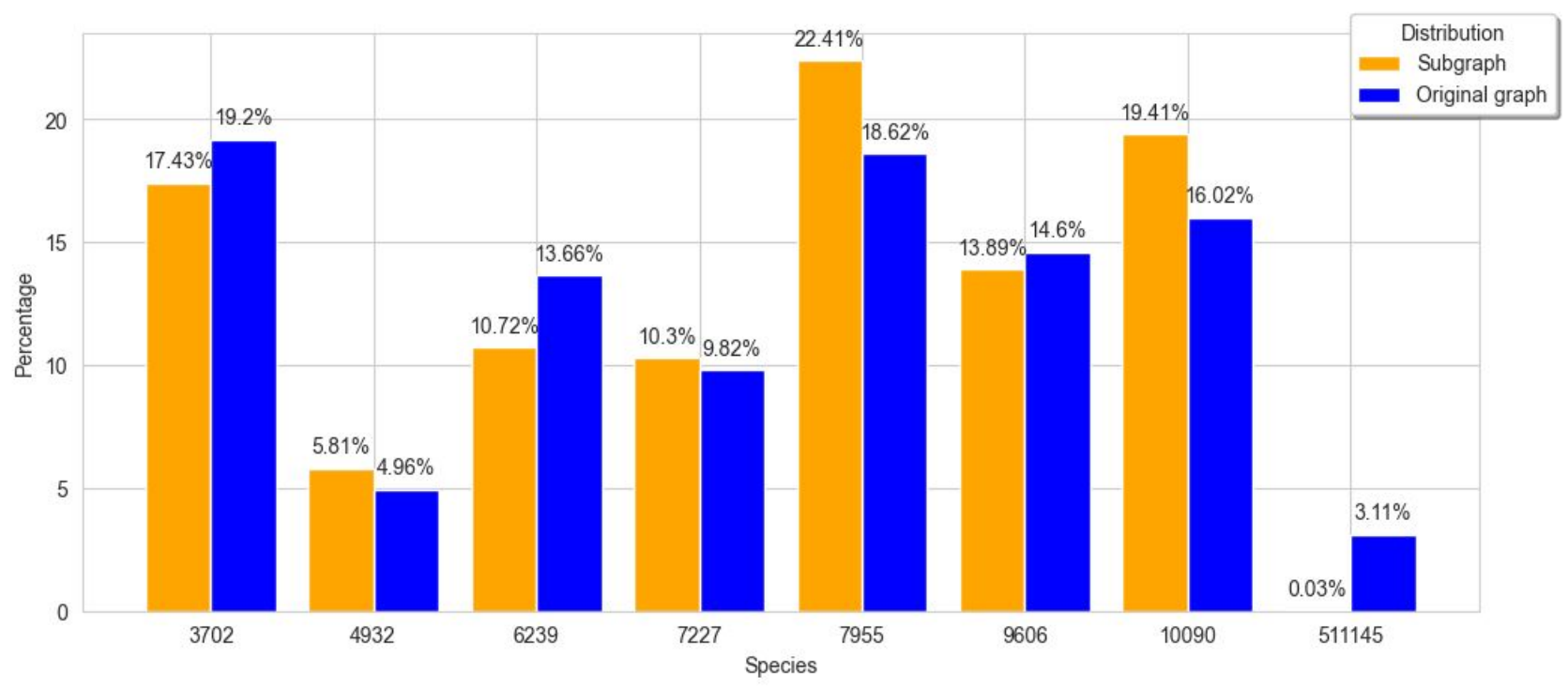


Figure: Node species distribution of a subgraph generated by Leiden with community size constraints compare to original graph

OVERLAPPING COMMUNITY CONSTRAINTS

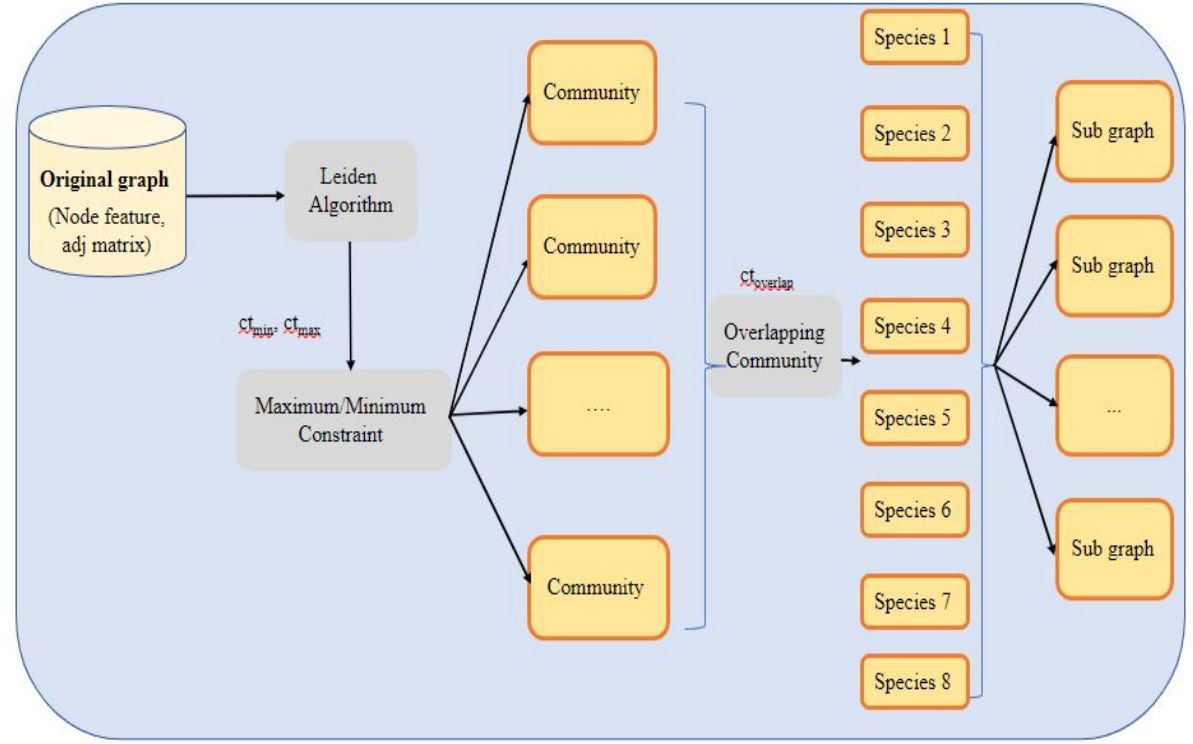


Figure: Leiden algorithm with community size and overlapping community constraints.

OVERLAPPING COMMUNITY CONSTRAINTS

Each community is assigned to one species S(C):

$$S(C) = \{i^*\} = \left\{ \underset{i \in S(\mathbf{G})}{\operatorname{argmax}} f_i^C \right\}$$

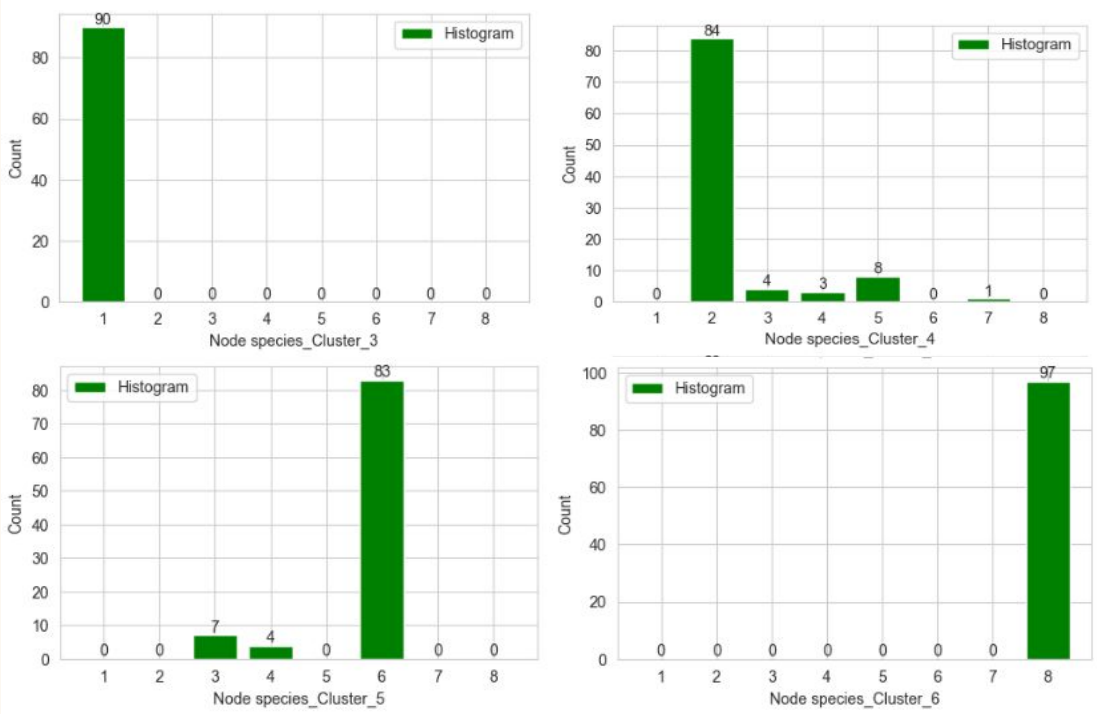


Figure: Communities with one species

OVERLAPPING COMMUNITY CONSTRAINTS

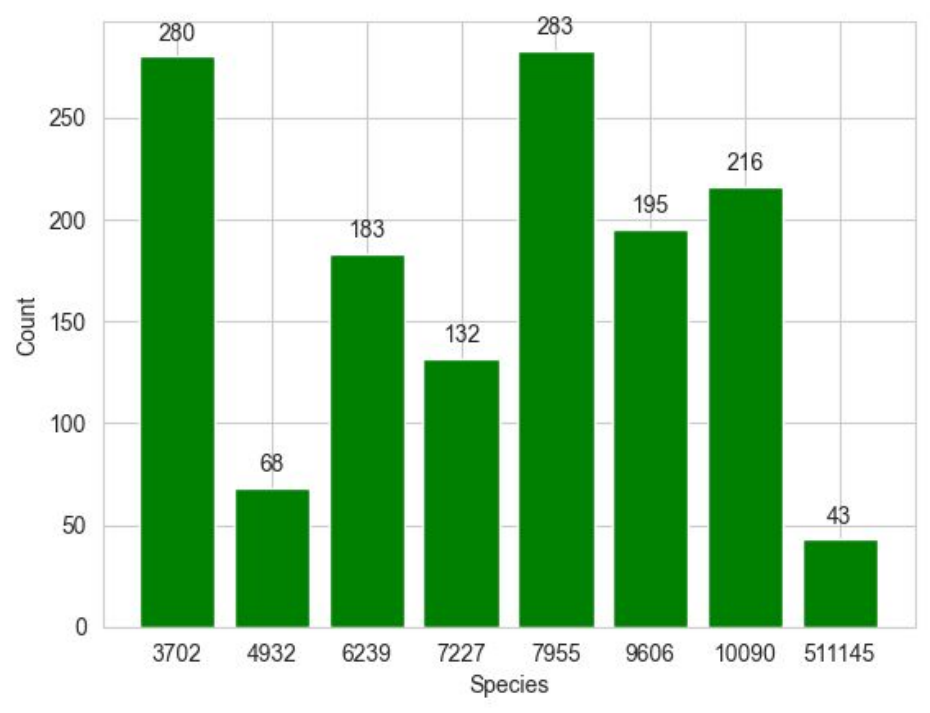


Figure: Number of communities in each species

OVERLAPPING COMMUNITY CONSTRAINTS

Each community is assigned to more than one species S(C):

$$S(C) = \{i | f_i^C \geq ct_{overlap} f_{i^*}^C\}$$

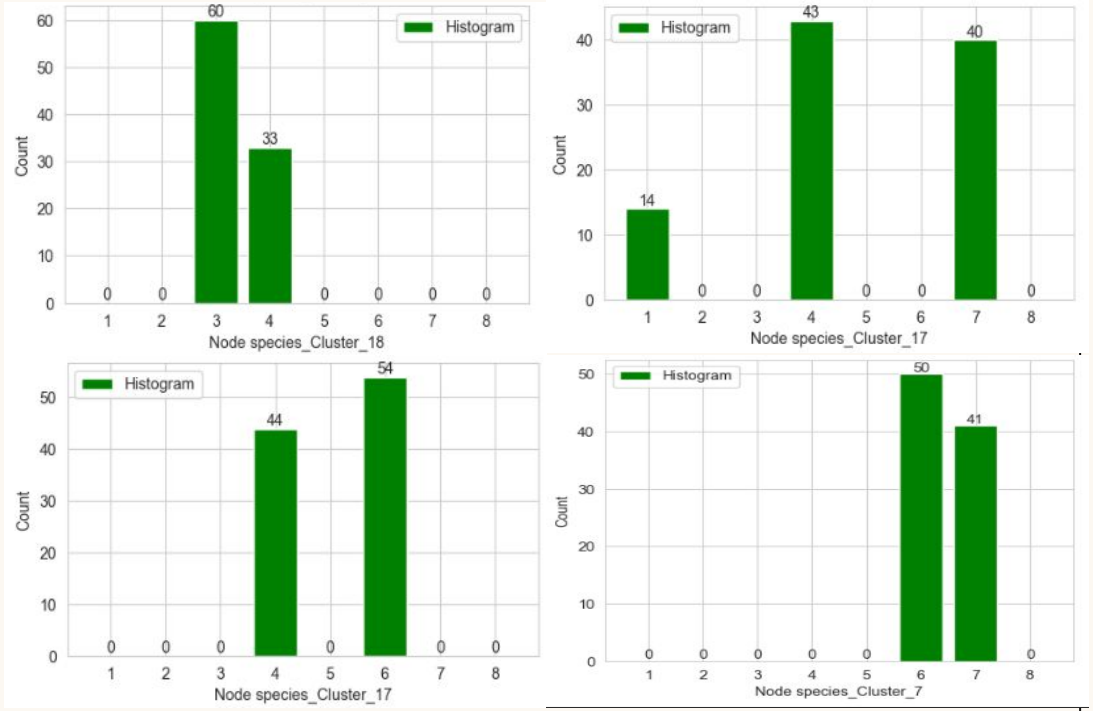


Figure: Communities with multiple species

OVERLAPPING COMMUNITY CONSTRAINTS

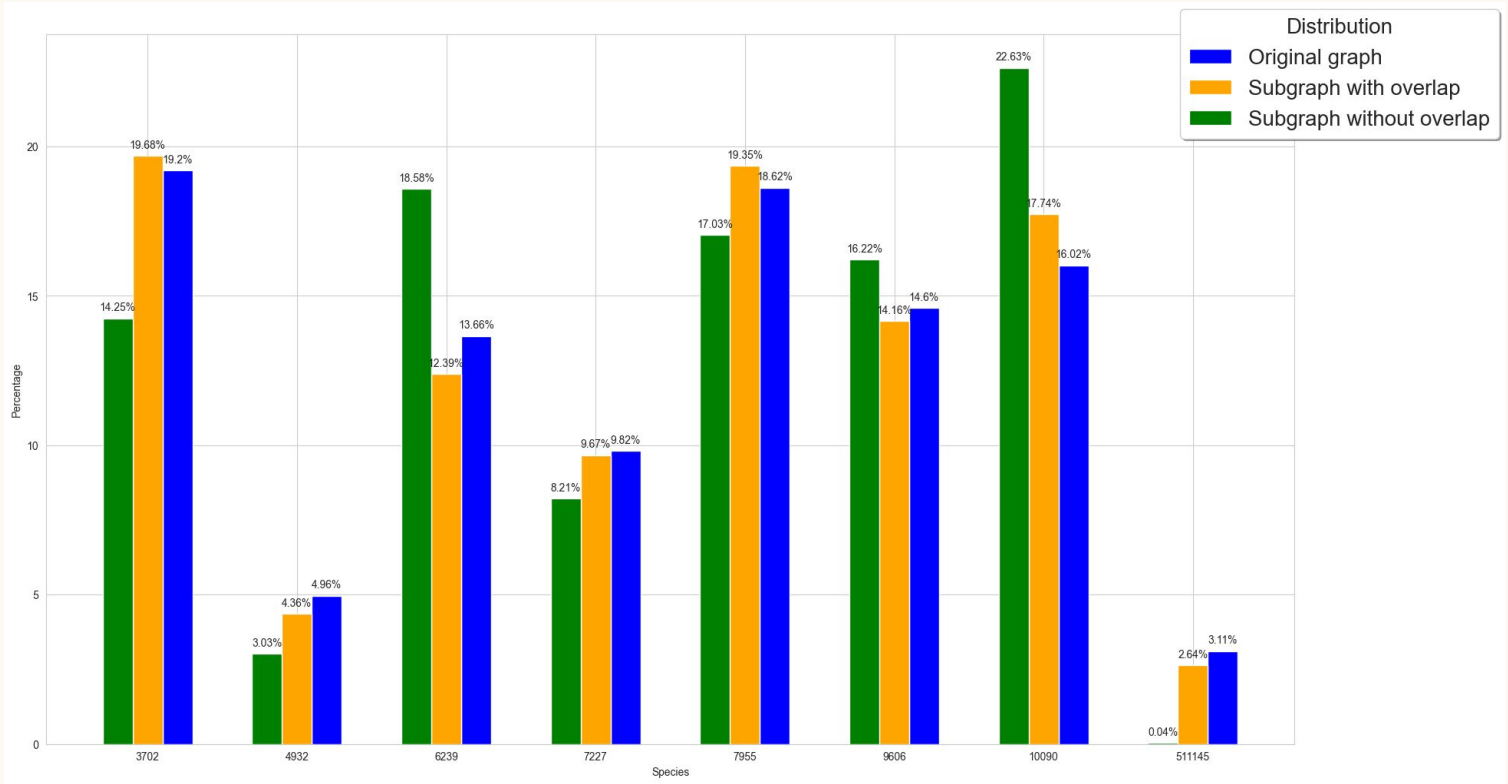


Figure: Node species distribution of a subgraph generated by Leiden with constraints compare to original graph

05 Experiments

Experiments setup and results

EXPERIMENTS SETUP

- Experiments are performed with *ogbn-proteins* dataset
- **r-Cluster-GCN**: Cluster-GCN with random partition.
- **LeidenGCN:** Cluster-GCN with *Leiden algorithm* instead of *METIS algorithm*.
- **b-LeidenGCN.** LeidenGCN with $\mathbf{ct}_{\mathbf{min}}$ and $\mathbf{ct}_{\mathbf{max}}$ constraints.
- **ob-LeidenGCN:** b-LeidenGCN with $\mathbf{ct}_{\mathbf{overlap}}$ constraint.

Hyperparameter	Value
Number of layers	{3,4,5,6,7}
Number of subgraphs	20
Edge feature learning type	Mean
Epochs	1000
Learning rate	0.005
Propagation dropout rate	0.5
Hidden channels	256

Table: Common hyperparameters between architectures

RESULTS

Algorithm	Modularity
Random	-0.00003675
METIS	0.08049441
Leiden	**0.73213373**
Leiden , $ct_{min} = 80$ and $ct_{max} = 100$	0.09024661
Leiden , $ct_{min} = 200$ and $ct_{max} = 300$	0.18143729
Leiden, $ct_{min} = 400$ and $ct_{max} = 500$	0.19769391

Table: Modularity comparison with different constraints value and different algorithms.

Algorithm	ROC-AUC
Cluster-GCN	0.7513 ± 0.0044
r-Cluster-GCN	**0,7771 ± 0,0025**
b-LeidenGCN (ours)	0.7630 ± 0.0030
ob-LeidenGCN (ours)	0.7746 ± 0.0007

Table: Performance comparison between models

RESULTS

	ROC-AUC				
Algorithm	3 layers	4 layers	5 layers	6 layers	7 layers
r-Cluster-GCN	**0.7771**	0.7833	0.7842	0.7643	0.7597
ob-LeidenGCN (ours)	0.7746	**0.7848**	**0.7940**	**0.7864**	**0.7780**

Table: Comparison of stacking deeper models

RESULTS

Algorithm	ROC-AUC
ob-LeidenGCN, $ct_{overlap} = 0.1$	0.7839313367
ob-LeidenGCN, $ct_{overlap} = 0.2$	0.7867266871
ob-LeidenGCN, $ct_{overlap} = 0.3$	0.7872546432
ob-LeidenGCN, $ct_{overlap} = 0.4$	0.788294836
ob-LeidenGCN, $ct_{overlap} = 0.5$	**0.7943720784**
ob-LeidenGCN, $ct_{overlap} = 0.6$	0.7875076298
ob-LeidenGCN, $ct_{overlap} = 0.7$	0.7875076298
ob-LeidenGCN, $ct_{overlap} = 0.8$	0.7910601422
ob-LeidenGCN, $ct_{overlap} = 0.9$	0.7913429585

Table: The effect of ctoverlap constraints.

06 Conclusion

- Summary of the thesis
- Future works

CONCLUSION

- **Better** than the **random partition**, **outperform** the **METIS** algorithm.
- Can **improve** model performance to some extent.

→ Can be the basis for building more complex and deeper architectures

FUTURE WORK:

- Extend to various community detection algorithms
- Experiment on more datasets of different fields to demonstrate the effect of adding constraints

Thanks for watching!

Please give us any question!