THE PRESENTATION

Presented by

EMKs

www.GraphsInclass.com :)

OUTLINE :

1. **Pre-intro -** (The problem - communities in networks)
2. **Intro (The hero) -** the original Louvain algorithm.
3. **The plot twist** - Louvain’s problems and why people complain.
4. **The Real work -** what are the 80 papers ?
5. **Lessons learned -** what are the 80 papers ?
6. **What if we have more time what would we do?**

Networks Are Everywhere

Networks are everywhere around us

- Social media
- Airports and flights
- Proteins in biology
- Web pages

All of these can be represented as networks

communities= clusters=bubbles=modules=collections

From Real Life to Graphs

How do we model this?

- Person → Node
- Relationship → Edge

If two people interact more, their connection is stronger.
This gives us a **graph representation.**

Measuring Relationships in a Graph

To measure the interaction between people in a network, we assign a number (weight) to each edge.

This number represents how strong or weak the relationship is.

For example:

A value of 1.0 means a very strong or good relationship.

A value like –0.5 can represent a weak or negative relationship.

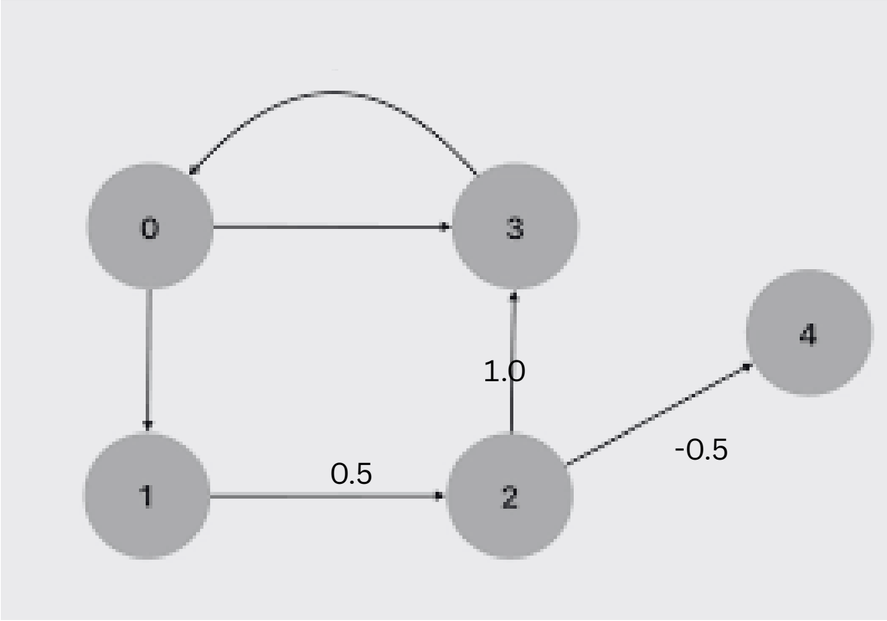
These values are based on assumptions, but they help us model real life.

We also assume the relationship is **symmetric**.

If person X has a relationship value of 0.7 with person Y, then Y also has a value of 0.7 with X.

If X likes Y, then Y likes X, and the same applies to **dislike**.

Now, instead of focusing on just two people, we look at many people together.

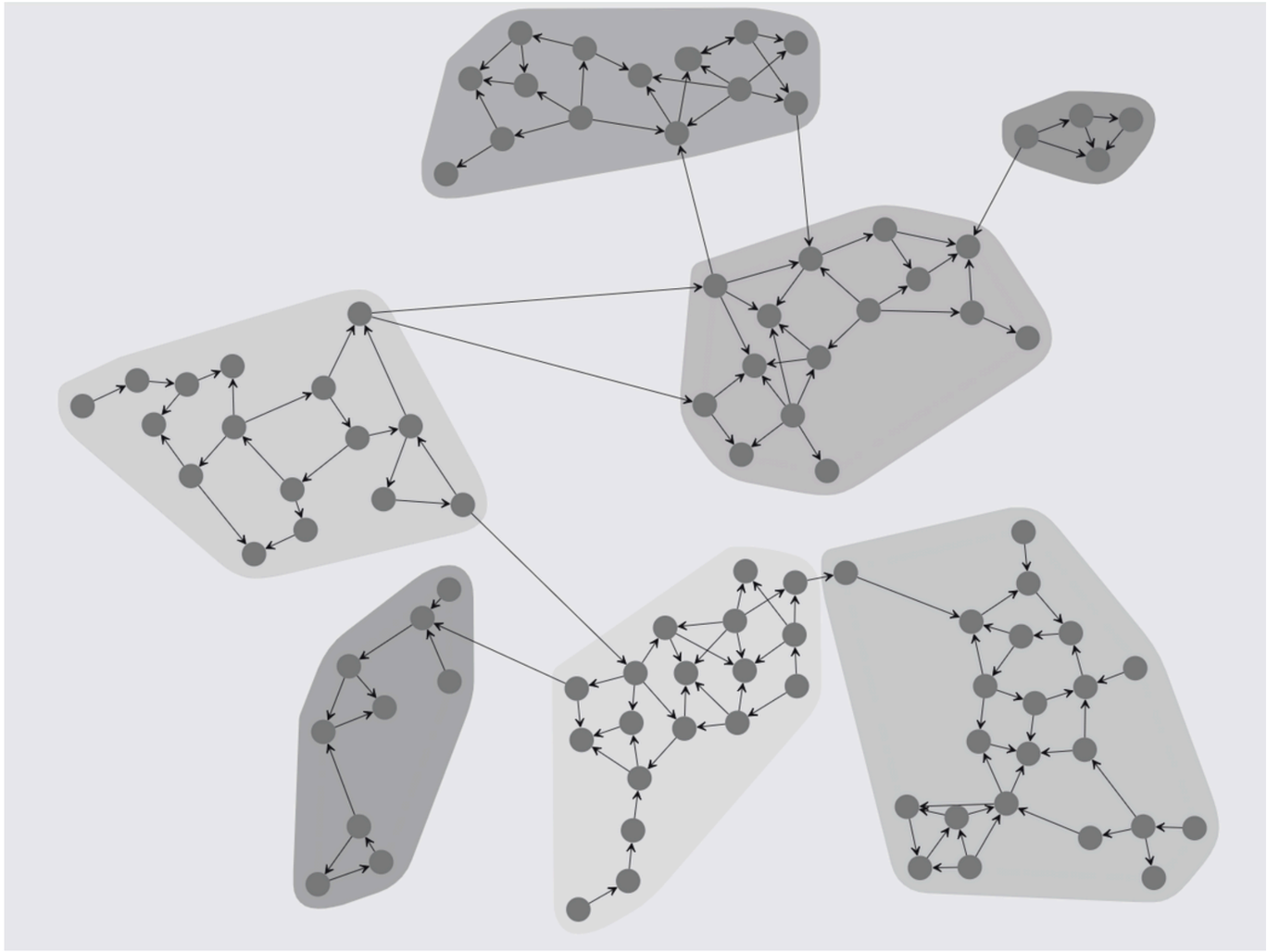


Graph Partitioning

What is grouping?
Grouping means partitioning a graph into sets of nodes such that :

- Connections inside groups are strong
- Connections between groups are weak
-

We can assign weights to edges to represent interaction strength.



Why Is This Hard?

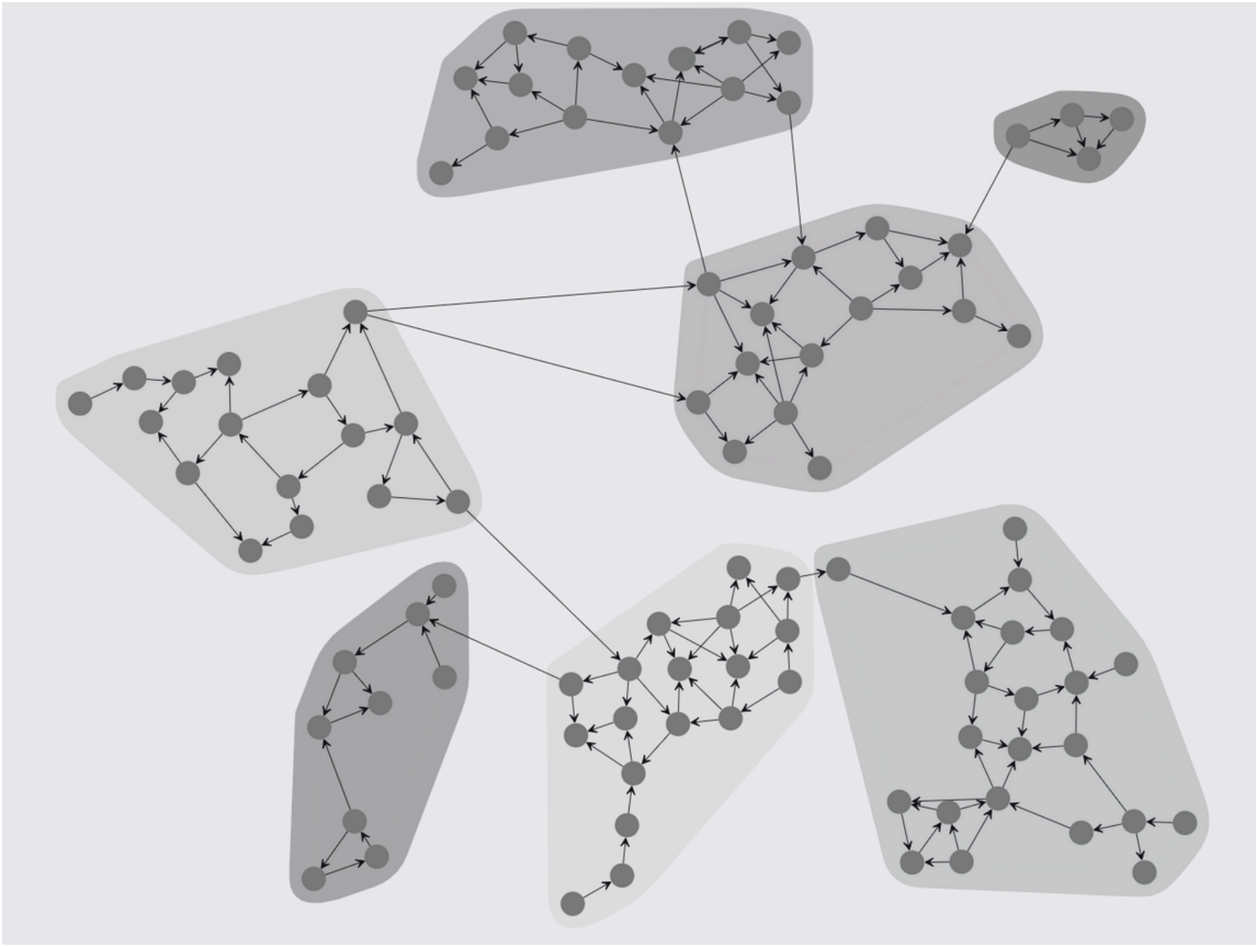
Many possible partitions
A single graph can be partitioned in many ways.

- 4 nodes → **15 partitions**

  So one possible partition might look like :
  C = { {x,y}, {z,u}, … } and this is just one partition.....

- 5 nodes → **52 partition**
- 6 nodes → **203 partitions**

so what do u think about a small number like 50 node ?



- 50 nodes → **1857242687710782704382577671819089174992218527 70** partitions ≈ 1.857×10^{47}

Real World ExamIpes:

YouTube. Videos are **nodes**, and **edges** represent similarity or co-watch behavior. The more related the videos, the stronger the edge. Communities here become ‘topic bubbles’. And that’s literally how you fall into a procrastination loop

searching for papers. Imagine papers are nodes, and edges represent similarity based on keywords, citations, or topic embeddings. When you search for something name, keywords, year the system tries to pull the most relevant community of papers and push away the weakly connected ones, like outdated or off-topic papers.

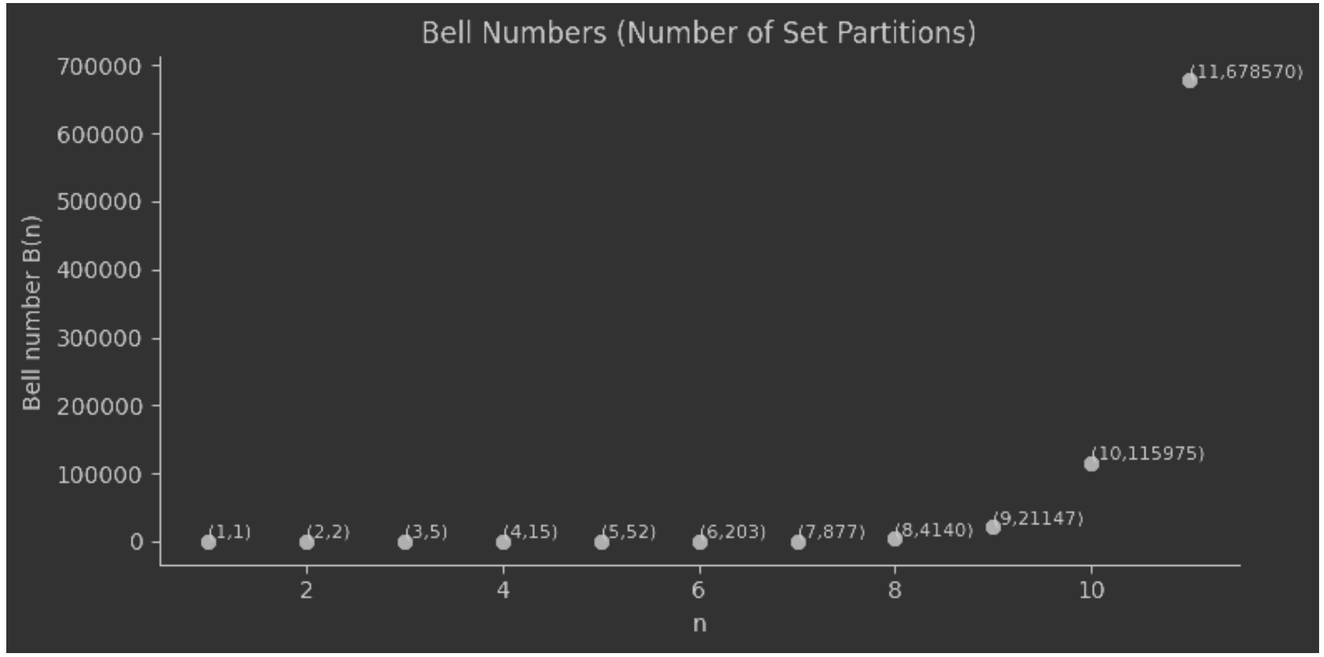
GPT, **DeepSeek**, **search engines** , all of them depend on building and organizing huge networks behind the scenes.

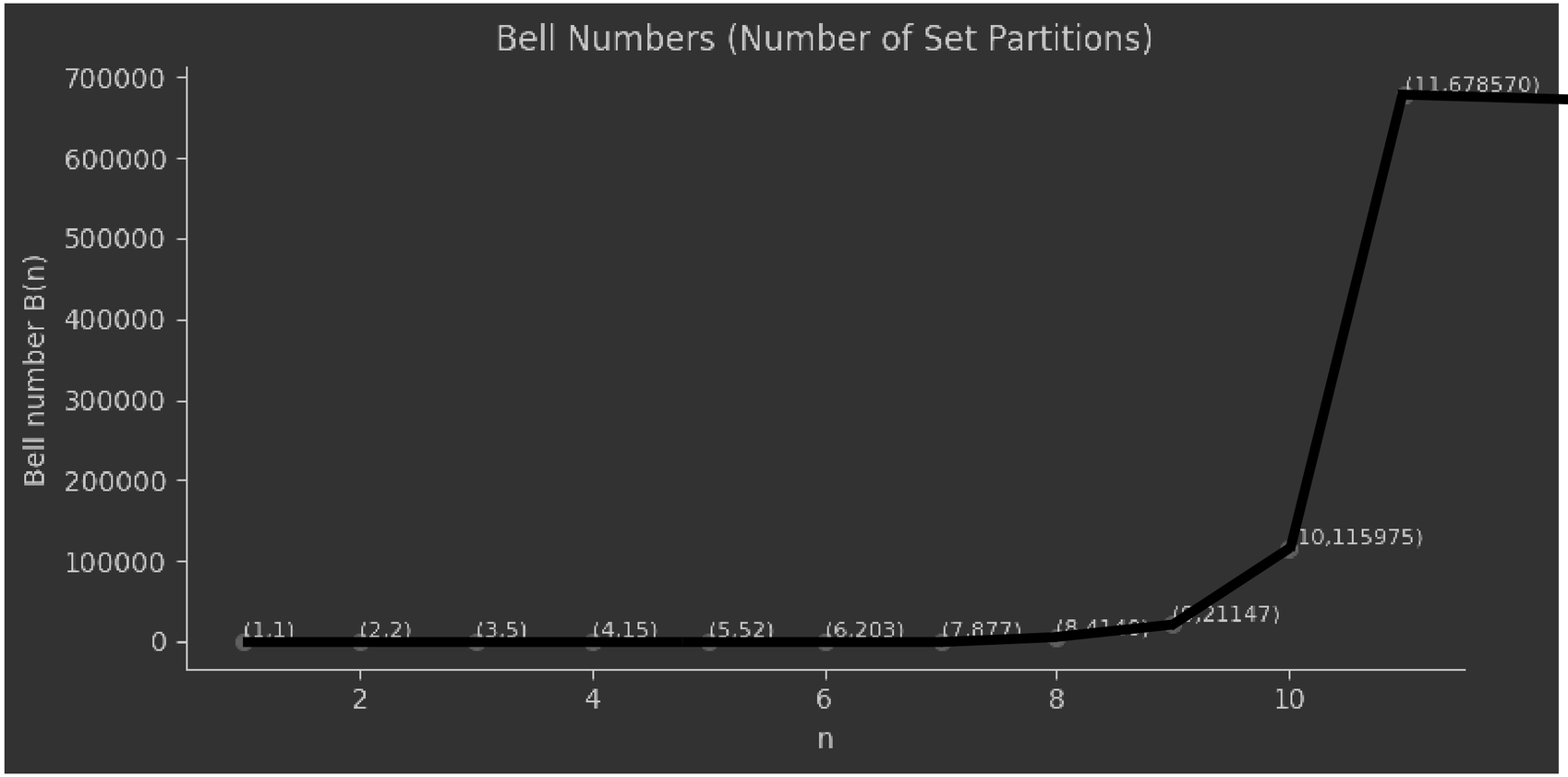
Obsidian graph view. If you’ve used it, you literally see clusters in your notes. which are personal knowledge networks, and communities show topics you keep linking together

Why We Need Algorithms

Brute force is not feasible
Checking all possible partitions is too **slow**.

We need an efficient method to find good communities.





We need an efficient method to find good communities.

Say welcom to

“**Louvain**” *

:)

www.LouvainToWorld.com :)

Louvain Algorithm

(“Louvain” , French origin)
A fast and efficient method for finding communities in networks.

Definition :
is the optimization of **modularity** as the algorithm progresses , or we can say it is a hierarchical **clustering** method for detecting community structures within networks. A community is defined as a subset of nodes with dense internal connections relative to sparse external connections.

Journal of Statistical Mechanics: Theory and Experiment
An IOP and SISSA journal

Fast unfolding of communities in large networks

J. Stat. Mech. (2008) P10008

Vincent D Blondel[1], Jean-Loup Guillaume[1,2], Renaud Lambiotte[1,3] and Etienne Lefebvre[1]

[1] Department of Mathematical Engineering, Université Catholique de Louvain, 4 avenue Georges Lemaitre, B-1348 Louvain-la-Neuve, Belgium
[2] LIP6, Université Pierre et Marie Curie, 4 place Jussieu, F-75005 Paris, France
[3] Institute for Mathematical Sciences, Imperial College London, 53 Prince’s Gate, South Kensington Campus, London SW7 2PG, UK
E-mail: vincent.blondel@uclouvain.be, jean-loup.guillaume@lip6.fr, r.lambiotte@imperial.ac.uk and pixetus@hotmail.com

Received 18 April 2008
Accepted 3 September 2008
Published 9 October 2008

Online at stacks.iop.org/JSTAT/2008/P10008
doi:10.1088/1742-5468/2008/10/P10008

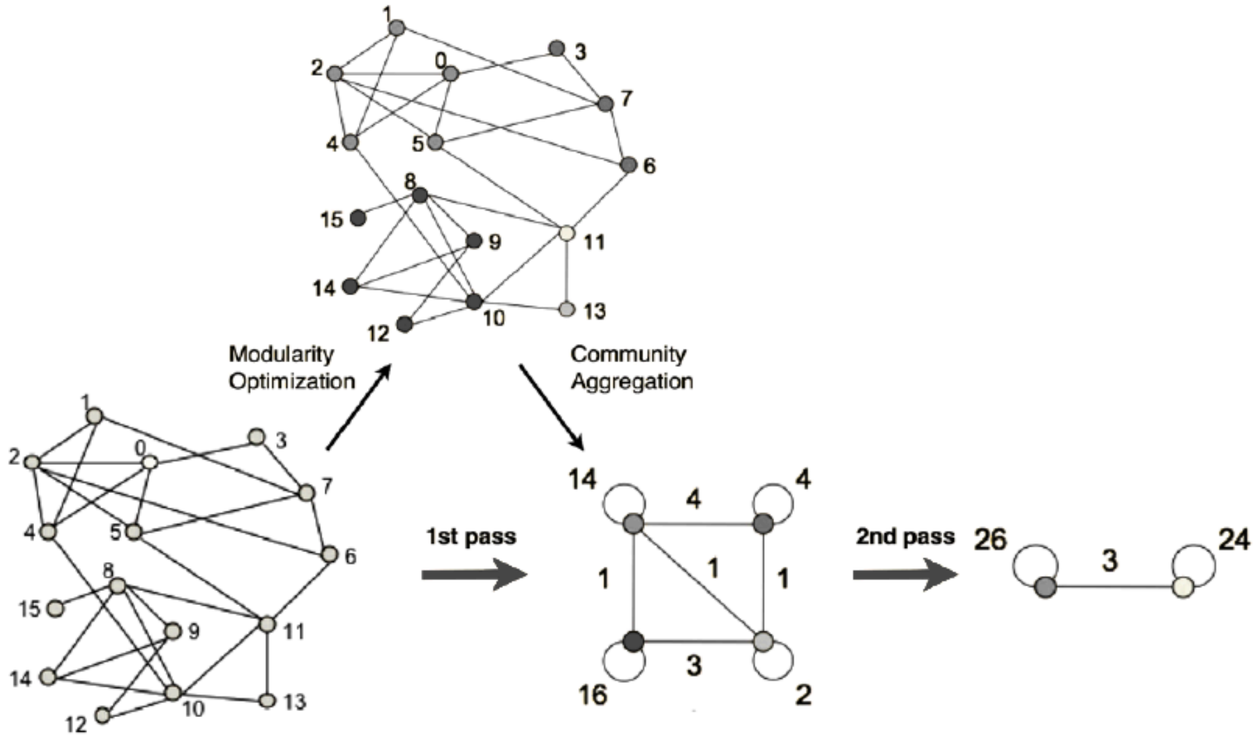
What the Heck does this mean :) ?

The Louvain algorithm is a smart shortcut for finding communities.

- **It does not check every possible grouping**
- **It makes local, greedy decisions**
- **It improves the result step by step**

Result:

- A clear map of the network’s real structure
- Very fast, even for large graphs



Louvain Algorithm

("Louvain" , French origin)
A fast and efficient method for finding communities in networks.

Definition :
is the optimization of **modularity** as the algorithm progresses , or we can say it is a hierarchical **clustering** method for detecting community structures within networks. A community is defined as a subset of nodes with dense internal connections relative to sparse external connections.

Journal of Statistical Mechanics: Theory and Experiment
An IOP and SISSA journal

Fast unfolding of communities in large networks

J. Stat. Mech. (2008) P10008

Vincent D Blondel[1], Jean-Loup Guillaume[1,2], Renaud Lambiotte[1,3] and Etienne Lefebvre[1]

[1] Department of Mathematical Engineering, Université Catholique de Louvain, 4 avenue Georges Lemaitre, B-1348 Louvain-la-Neuve, Belgium
[2] LIP6, Université Pierre et Marie Curie, 4 place Jussieu, F-75005 Paris, France
[3] Institute for Mathematical Sciences, Imperial College London, 53 Prince's Gate, South Kensington Campus, London SW7 2PG, UK
E-mail: vincent.blondel@uclouvain.be, jean-loup.guillaume@lip6.fr, r.lambiotte@imperial.ac.uk and pixetus@hotmail.com

Received 18 April 2008
Accepted 3 September 2008
Published 9 October 2008

Online at stacks.iop.org/JSTAT/2008/P10008
doi:10.1088/1742-5468/2008/10/P10008

What is the modularity ?!

Modularity measures:

- How strong the connections are inside communities
- Compared to connections between communities

High modularity means:
Dense connections inside groups
Sparse connections between groups

which basically shows : “**Are these communities meaningful or just random?**”

Modularity has been used to compare the quality of the partitions obtained by different methods, but also as an objective function to optimize [13]. Unfortunately, exact modularity optimization is a problem that is computationally hard [14] and so approximation algorithms are necessary when dealing with large networks. The fastest approximation algorithm for optimizing modularity on large networks was proposed by Clauset *et al* [8]. That method consists in recurrently merging communities that optimize the production of modularity. Unfortunately, this greedy algorithm may produce values of modularity that are significantly lower than what can be found by using, for instance, simulated annealing [15]. Moreover, the method proposed in [8] has a tendency to produce

We now introduce our algorithm that finds high modularity partitions of large networks in a short time and that unfolds a complete hierarchical community structure for the network, thereby giving access to different resolutions of community detection. Contrary

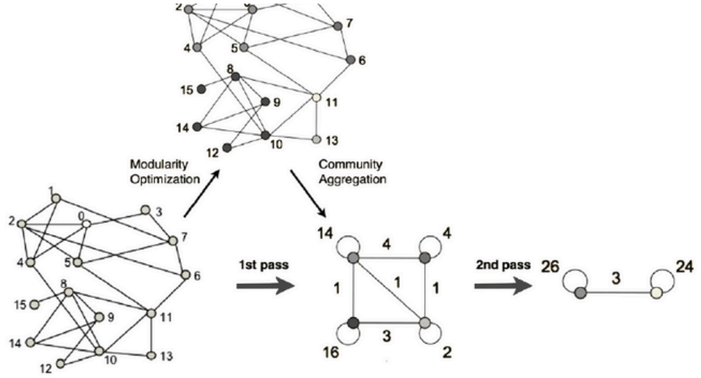


Figure 1. Visualization of the steps of our algorithm. Each pass is made of two phases: one where modularity is optimized by allowing only local changes of communities; one where the communities found are aggregated in order to build a new network of communities. The passes are repeated iteratively until no increase of modularity is possible.

J. Stat. Mech. (2008) P10008

are based on the maximization of an objective function [8]–[10]. The quality of the partitions resulting from these methods is often measured by the so-called modularity of the partition. The modularity of a partition is a scalar value between −1 and 1 that measures the density of links inside communities as compared to links between communities [5, 11]. In the case of weighted networks (weighted networks are networks that have weights on their links, such as the number of communications between two mobile phone users), it is defined as [12]

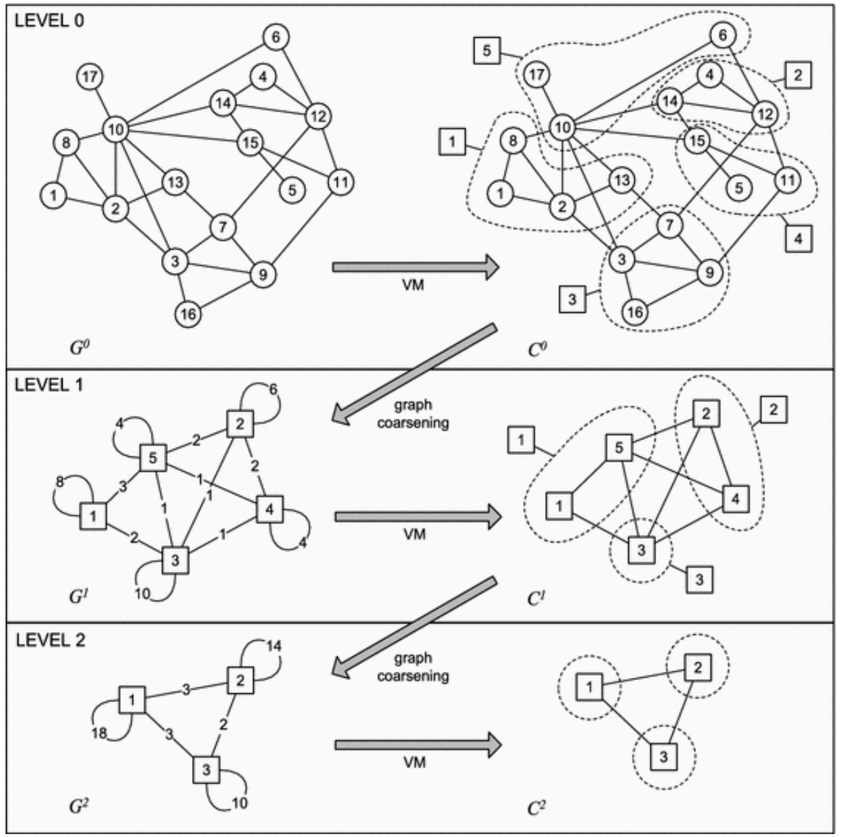
How Louvain Works?

Louvain works in two repeating phases:

Phase 1 – Local Moving

- Each node starts alone
- Nodes move to neighboring communities
- Only if modularity increases

Stop when no move improves the score.



Phase 2 – Aggregation

Each community becomes a super-node

Build a new, smaller graph

Repeat Phase 1 again

Pros, Cons, and Today’s View

Advantages (same with here

- Very fast (near-linear time)
- Scales to millions of nodes
- No brute force search

Disadvantages (maybe we change some stuff here)

- Resolution limit (small communities may disappear)
- Communities may be weakly connected
- Only non-overlapping communities

The Plot Twist
Louvain is good.....
but not perfect

why not perfect?

If Louvain is such a strong algorithm,
why do we have 80+ papers about it?

- **most papers say: “Louvain works well, but…” does it really is ?**

Most research focuses on four problem areas:

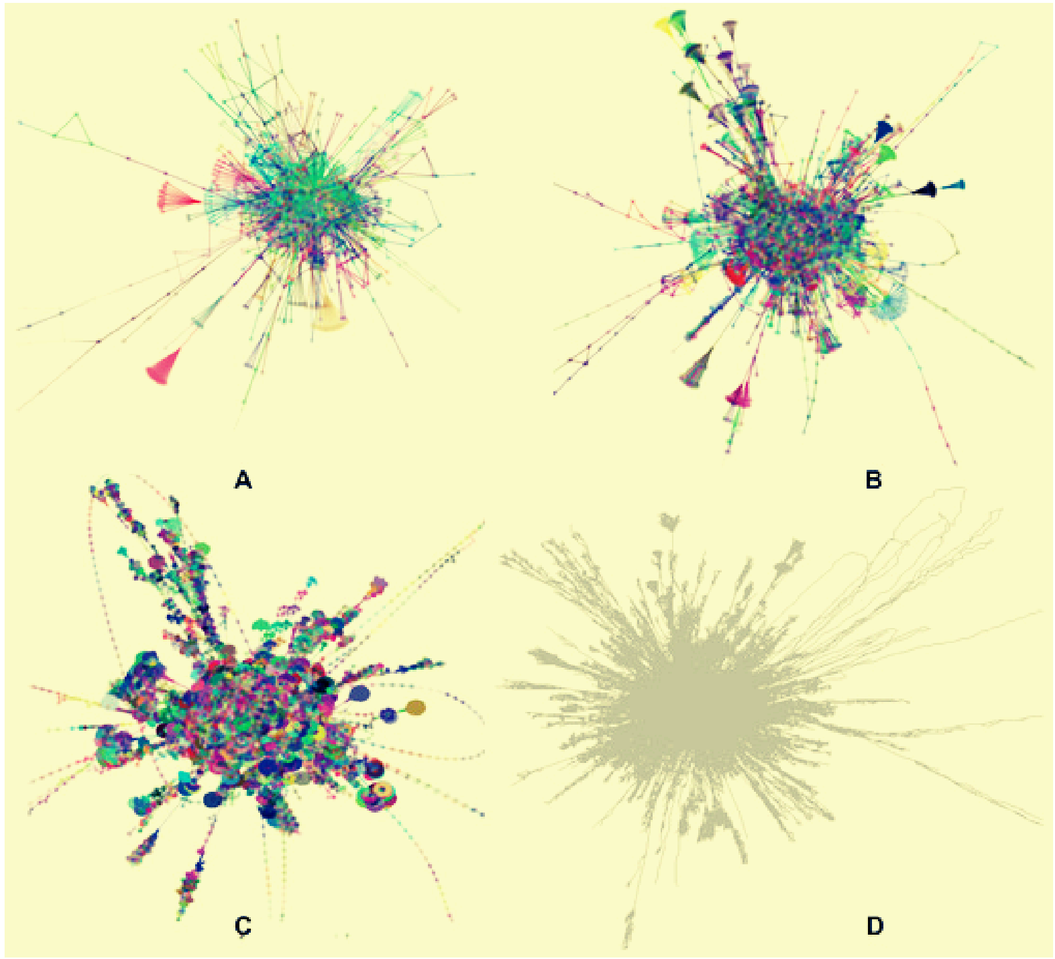
- **Scalability & parallelism**
- **Quality & resolution limit**
- **Dynamic / temporal networks**
- **Real-world applications**

Scalability & parallelism

Louvain is an efficient algorithm, but as the size of the graph increases **(millions or billions of edges and vertexes)**, its performance can degrade.

This is because it processes the graph sequentially, leading to slower execution times for very large datasets.

For **large-scale graphs**, this sequential nature can be a bottleneck, limiting the algorithm's scalability. The challenge lies in ensuring that Louvain remains effective without compromising on speed or the quality of the community detection process.



Influencer Detection in Social Networks

Traditional community detection algorithms like Louvain may struggle with tasks such as identifying influencers in large social networks, often leading to excessive fragmentation of communities.

Lack of Stability (çıkartabiliirz)

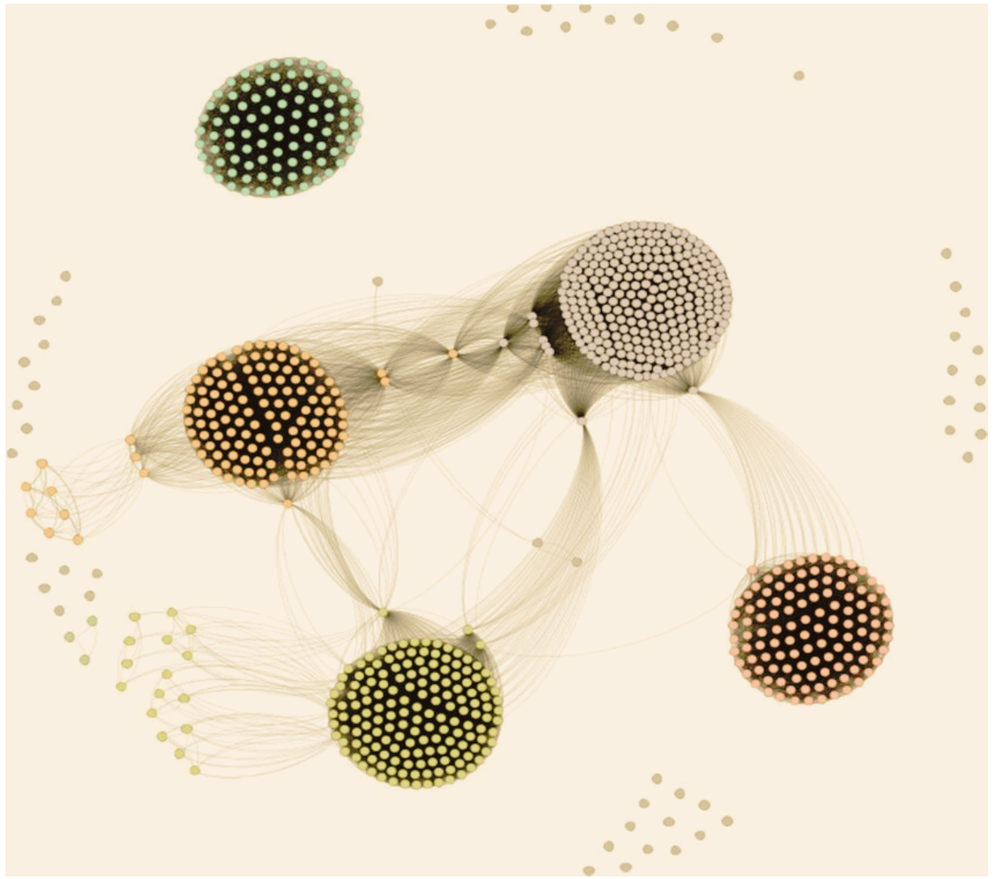
Louvain is non-deterministic. The algorithm can produce different results on different runs because of its reliance on random initialization. This makes the results inconsistent, which can be problematic in experiments or when reproducibility is essential.

Sensitivity to Resolution Limit (çıkartabilirz)

Louvain suffers from the resolution limit problem, where it tends to merge smaller communities into larger ones, especially in large networks. This is problematic when trying to detect smaller-scale communities, as the algorithm may fail to identify them.

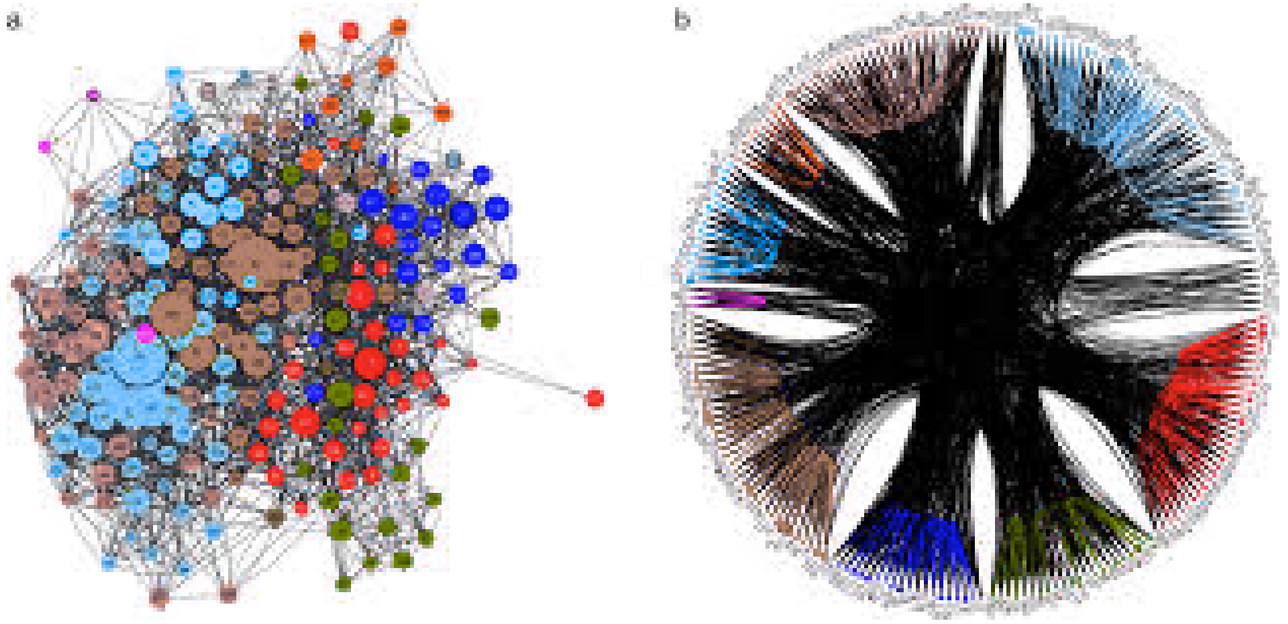
Modularity and Processing Time Efficiency

Louvain’s performance can be suboptimal in terms of modularity (quality of community structures) and processing time, especially when applied to large datasets or complex networks.



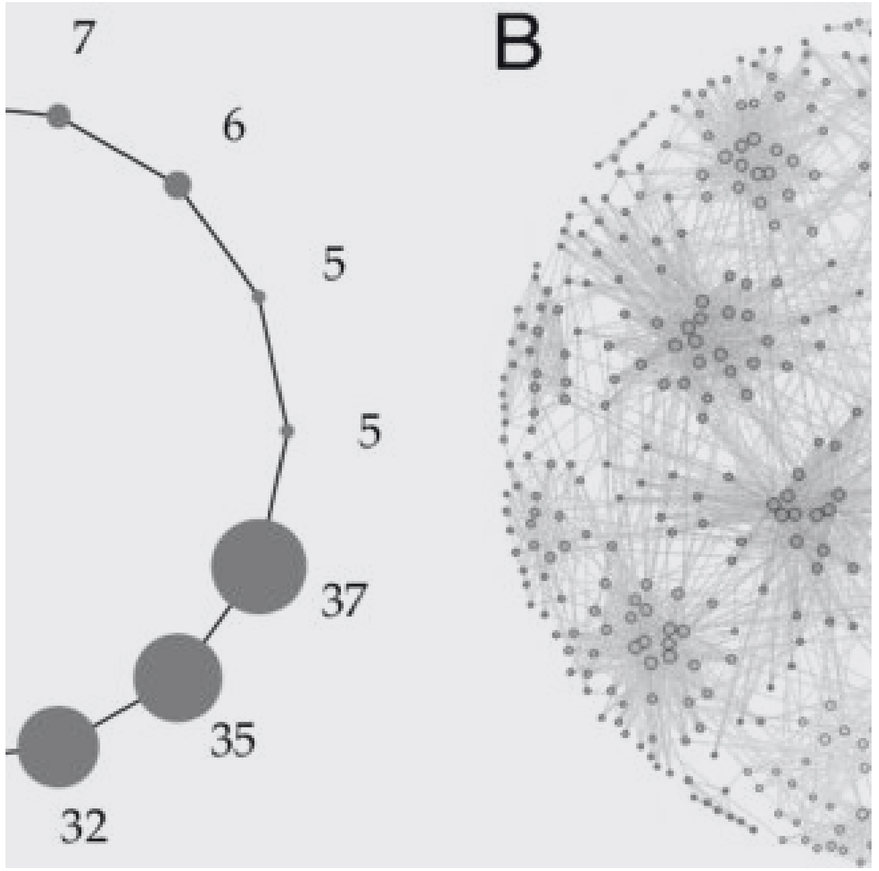
Incorporating Edge Weights into Community Detection

The original Louvain algorithm does not consider edge weights, potentially overlooking the strength of connections between nodes, which can lead to less accurate community detection, especially in heterogeneous networks.



Sensitivity to Noise in Community Detection

In large and noisy networks, weak or insignificant edges can distort the community detection process, leading to inaccurate or irrelevant community partitions.



Computational Efficiency for Large Networks

While Louvain is effective, its execution time can be high for large networks, making it less suitable for real-time or large-scale applications.

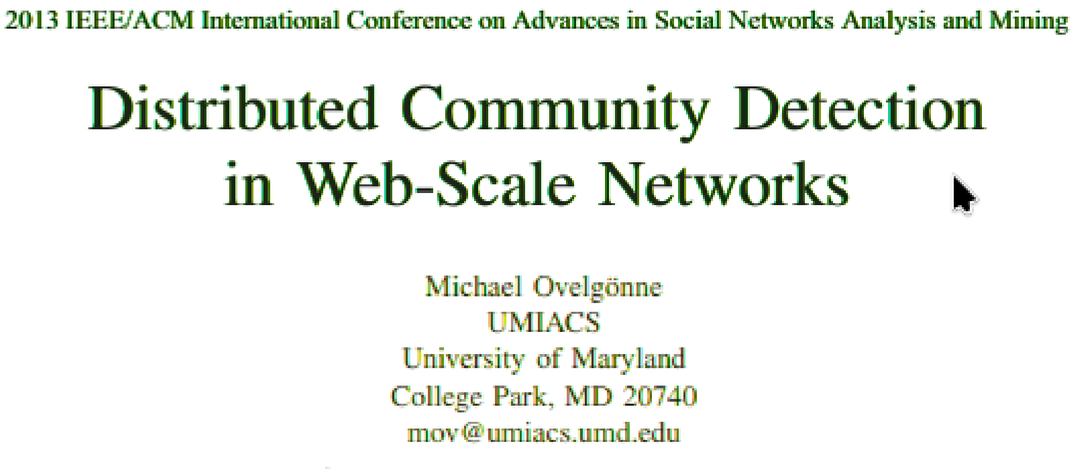
The spin-offs

what are the 80 papers ?

Community Detection in Large Networks

Community detection is essential for understanding large-scale networks.
Existing algorithms struggle with networks that have billions of edges.

Challenge: **Efficient and scalable methods are needed for huge graphs.**



2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining

Distributed Community Detection in Web-Scale Networks

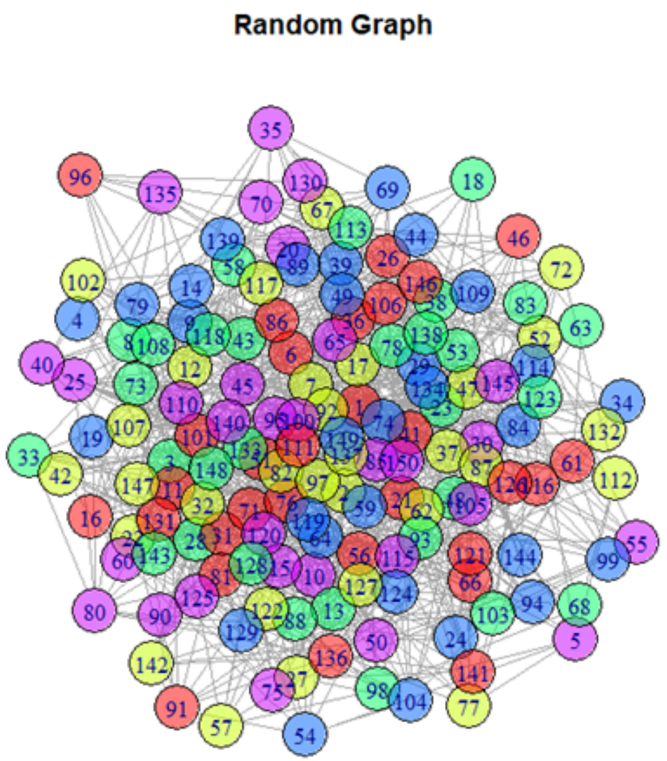
Michael Ovelgönne
UMIACS
University of Maryland
College Park, MD 20740
mov@umiacs.umd.edu

Source: [1]

Distributed Community Detection Approach

To address the scalability challenges posed by large graphs, distributed community detection methods are required. The study introduces a distributed approach that leverages parallelism to process large-scale networks efficiently. By splitting the workload across multiple machines, the algorithm can handle large networks more effectively than traditional sequential methods.

Problem: Community detection becomes computationally expensive as graph size increases.
Solution: Distributed processing allows for faster execution, making it feasible to process networks with billions of edges.



Core Groups via Ensemble Learning

The paper introduces a core group idea to make community detection more reliable on huge networks. Instead of trusting one single partition, it generates an ensemble of partitions and looks for nodes that repeatedly end up together.

Maximal Overlap: Nodes that appear in the same community across many partitions are treated as strong evidence of a true group.

Core Group Creation: These high-agreement nodes are merged into core groups, which act as stable, high-confidence building blocks before running the final community detection step.

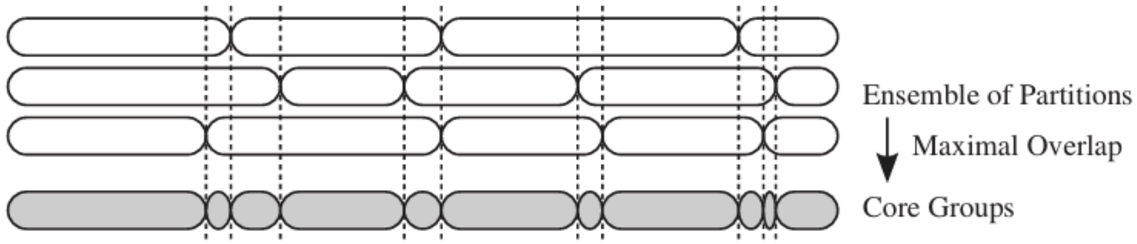


Fig. 1. In a core groups partition (bottom row) those vertices are in one group that are in the same community in all partitions of the ensemble (3 top rows).

Source: [1,Fig1]]

Label Propagation Step (How core groups are created)

The slide explains the label propagation step used to build core groups before running Louvain. Instead of giving each node a single label, the method runs many label propagations in parallel by storing a vector of labels per node. Over several propagation steps, each node updates its labels based on its neighbors, so labels spread and gradually stabilize in densely connected areas. After this process, nodes that repeatedly end up together across instances are grouped into core groups, which are high-confidence community building blocks. The main benefit is that these core groups compress the graph, making the final community detection step much more scalable.

Algorithm 1: Hadoop Core Group Identification

input : edgelist file $input\_file$, # of propagation steps p, # of label instances k
output: vertex-core group mapping file cg_file, edgelist file el_file

1 $passed\_labels \leftarrow$ `ReadMap` ($input\_file$, k)
2 **for** $i \leftarrow 1$ **to** $p - 1$ **do**
3 $\quad new\_labels \leftarrow$ `PropagateReduce` ($passed\_labels$)
4 $\quad passed\_labels \leftarrow$ `PropagateMap` (new_labels)
5 $final\_labels \leftarrow$ `CoreGroupsExportReduce` (new_labels, cg_file)
6 $final\_labels \leftarrow$ `EdgeListPreparationMap` ($final\_labels$)
7 $core\_group\_links \leftarrow$ `EdgeListPreparationReduce` ($final\_labels$)
8 $core\_group\_links \leftarrow$ `EdgeListExportMap` ($core\_group\_links$)
9 `EdgeListExportReduce` ($core\_group\_links$, el_file)

Source: [1,Algorithm 1]]

Scalability of Core Group Generation

The method builds core groups using repeated label propagation runs in parallel.
Figure 6 shows how the runtime of core group generation scales as the number of edges increases. Runtime grows in a predictable, roughly linear way, which supports scalability. Increasing the number of label instances (10, 20, 40) increases runtime, showing the cost–quality trade-off.

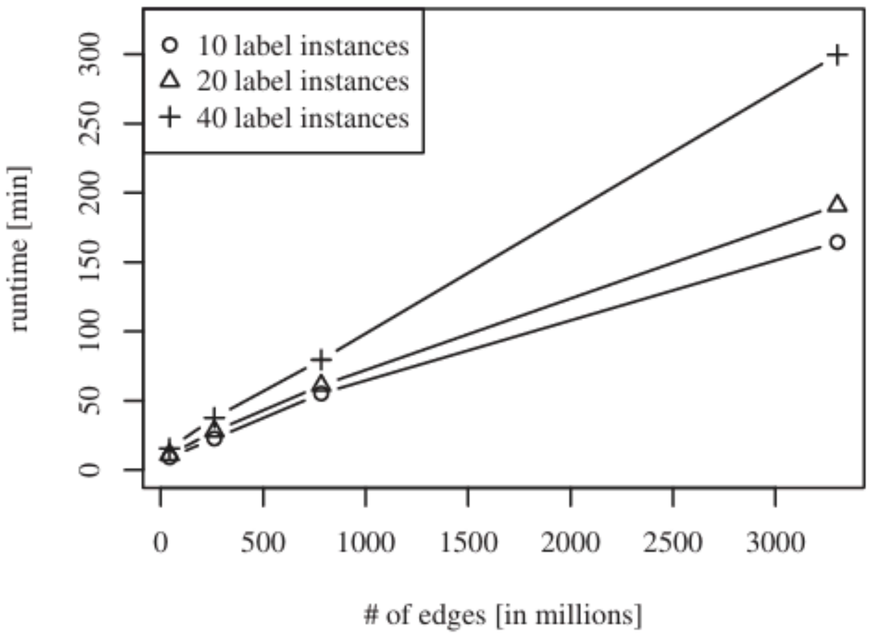
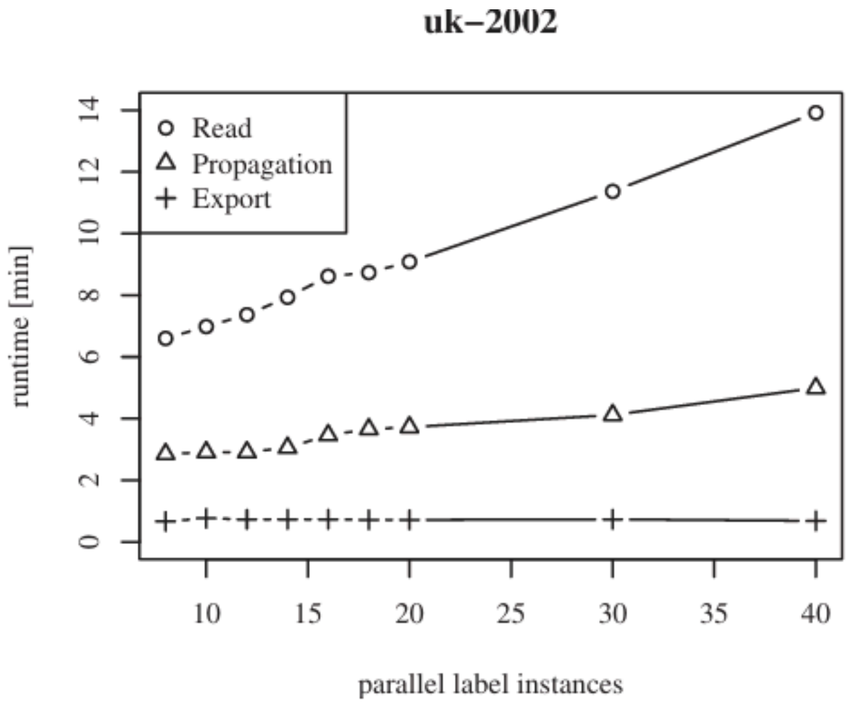


Fig. 6. Runtime of core group generation depending number of edges of graph for three different parameter values for the number of label instances. Additional to the two datasets used throughout the evaluation section we used a symmetrized version of the dataset *LiveJournal* [1] with ∼43 million edges and the dataset *uk-2005* [3] with ∼0.9 billion edges.

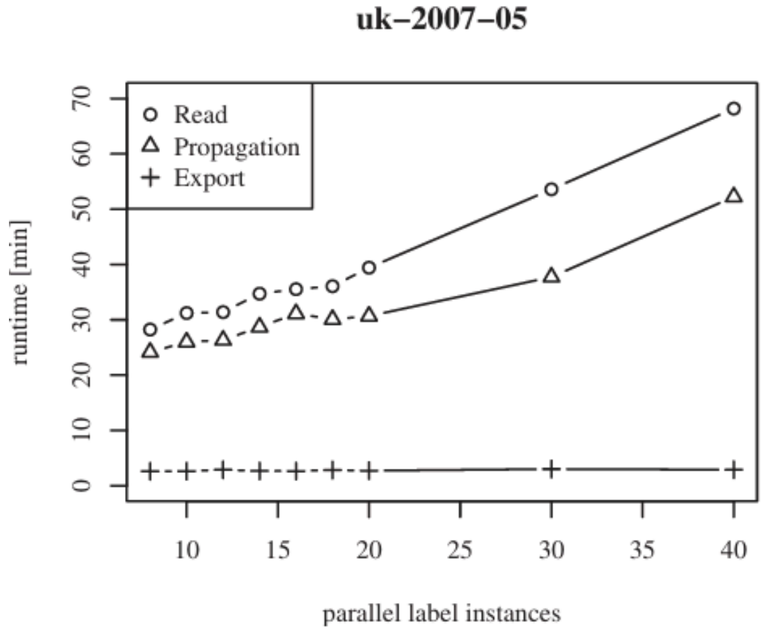
Source: [1,Fig6]]

Parallelism in Hadoop for Scalability

Hadoop is used to implement the distributed approach, enabling parallel processing of community detection tasks across multiple machines. Figure 4 shows that as the number of label instances increases, total runtime increases in a predictable, roughly linear way. The propagation stage is the dominant cost, while export remains small, which supports scalability on very large graphs.



Source: [1,Fig4]]



Experimental Results and Discussion

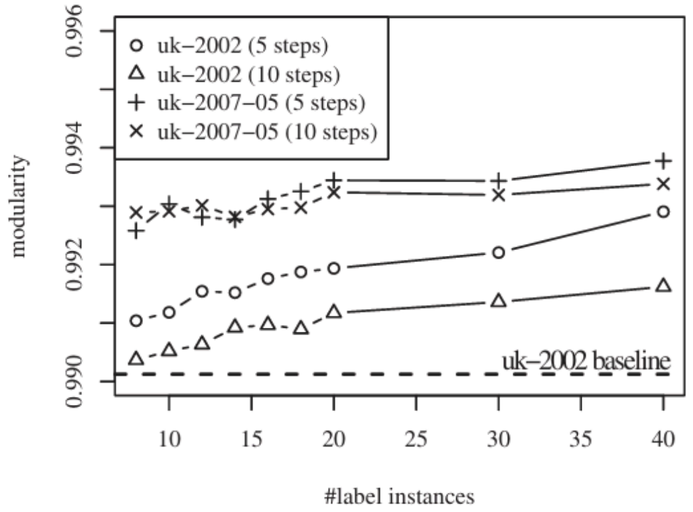
The experiments evaluate how the distributed core-group approach behaves as the graphs and parameters grow. The key result is scalability: runtime increases in a predictable, roughly linear way as the number of edges increases.

Scalability (Figure 6): As graph size grows (#edges in millions), runtime rises steadily.

Increasing the number of label instances (10 → 20 → 40) increases runtime, showing the cost of stronger evidence for core groups.

Quality (from Figure 5 in the paper): Despite the scalability focus, the method maintains high modularity, meaning community quality is not sacrificed.

Parallel behavior (from Figure 4 in the paper): Hadoop runtime grows predictably with more parallel label instances, supporting distributed execution at web scale.



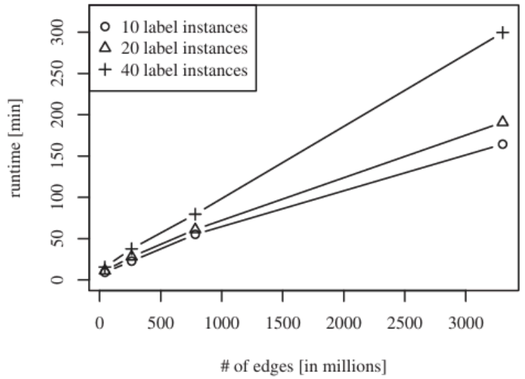


Fig. 6. Runtime of core group generation depending number of edges of graph for three different parameter values for the number of label instances. Additional to the two datasets used throughout the evaluation section we used a symmetrized version of the dataset *LiveJournal* [1] with ∼43 million edges and the dataset *uk-2005* [3] with ∼0.9 billion edges.

Source: [1,Fig6,Fig5]]

Conclusion and Future Work

The proposed distributed community detection algorithm offers a scalable solution for processing large networks. By leveraging parallelism and core group detection, the algorithm efficiently handles graphs with billions of edges.

Future Work will focus on improving the algorithm's parallelism and testing it on different types of networks to further optimize its performance. There is also potential for incorporating other distributed computing frameworks to enhance scalability.

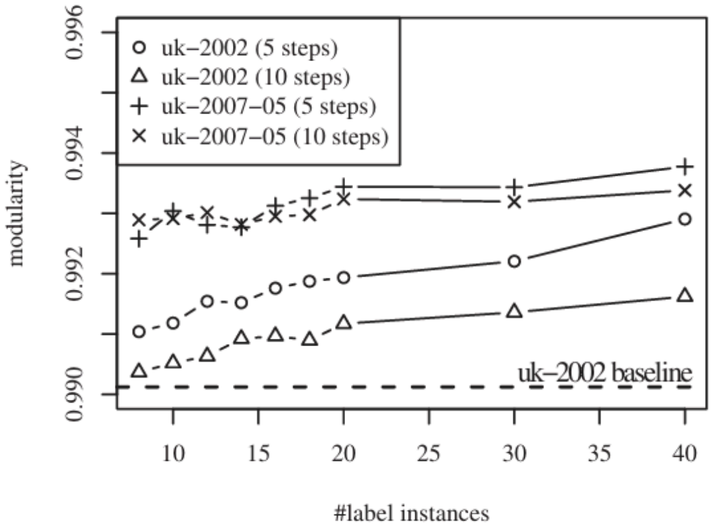


Fig. 5. Average modularity of 10 runs of the algorithm [2] of the graphs induced by core groups identified for settings of the number of label instances and the number of label propagation steps. For the smaller dataset *uk-2002* the average modularity identified without pre-processing is given as a baseline. For the larger dataset *uk-2007-05* starting the modularity optimization algorithm without pre-processing was not feasible because of memory limitations.

Source: [1,Fig5]]

Introduction to the Study

This 2025 study proposes influencer detection on Twitter (X) by combining Leiden Coloring (Leiden + graph coloring) with Degree Centrality.

Data: tweets crawled using the keyword “GarudaIndonesia” (22,623 rows). The evaluation uses two random samples: 1000 rows and 2000 rows.

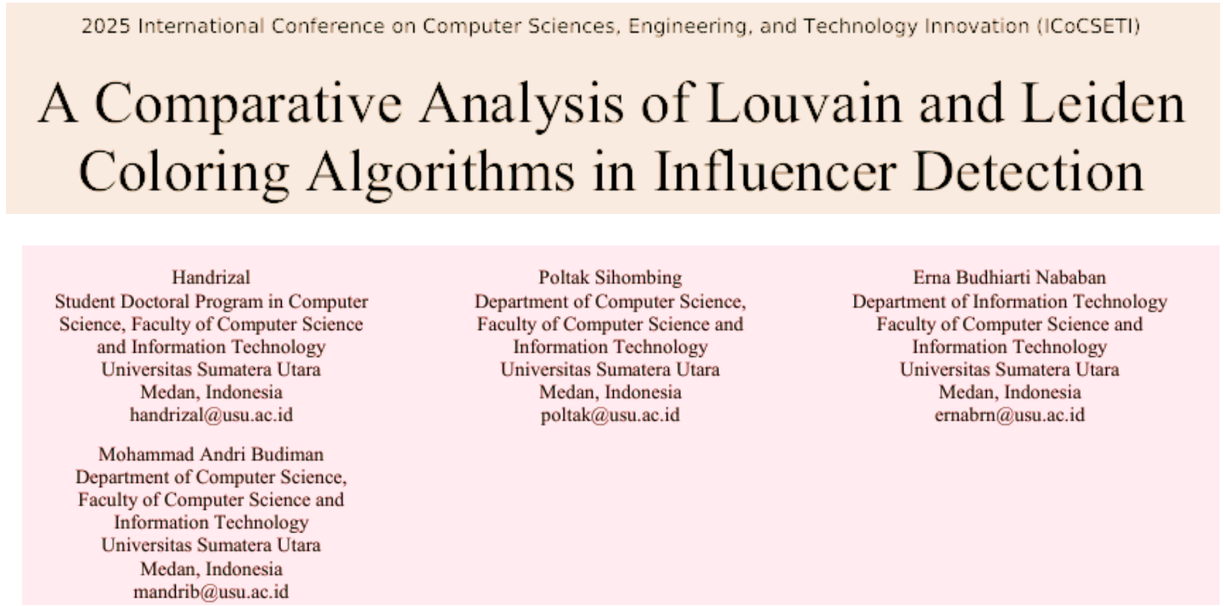
Goal: Compare Leiden Coloring vs. Louvain Coloring using modularity (community quality), processing time, number of communities, and top-influencer ranking.

Context: Louvain struggles with large graphs (e.g., Twitter) in a reasonable time.

Importance: Fast influencer detection is essential for marketing and content distribution.

Goal: Compare Louvain vs. Leiden based on:

- Modularity (community quality),
- Processing time (efficiency),
- Number of communities (granularity).



2025 International Conference on Computer Sciences, Engineering, and Technology Innovation (ICoCSETI)

A Comparative Analysis of Louvain and Leiden Coloring Algorithms in Influencer Detection

Handrizal
Student Doctoral Program in Computer Science, Faculty of Computer Science and Information Technology
Universitas Sumatera Utara
Medan, Indonesia
handrizal@usu.ac.id

Poltak Sihombing
Department of Computer Science, Faculty of Computer Science and Information Technology
Universitas Sumatera Utara
Medan, Indonesia
poltak@usu.ac.id

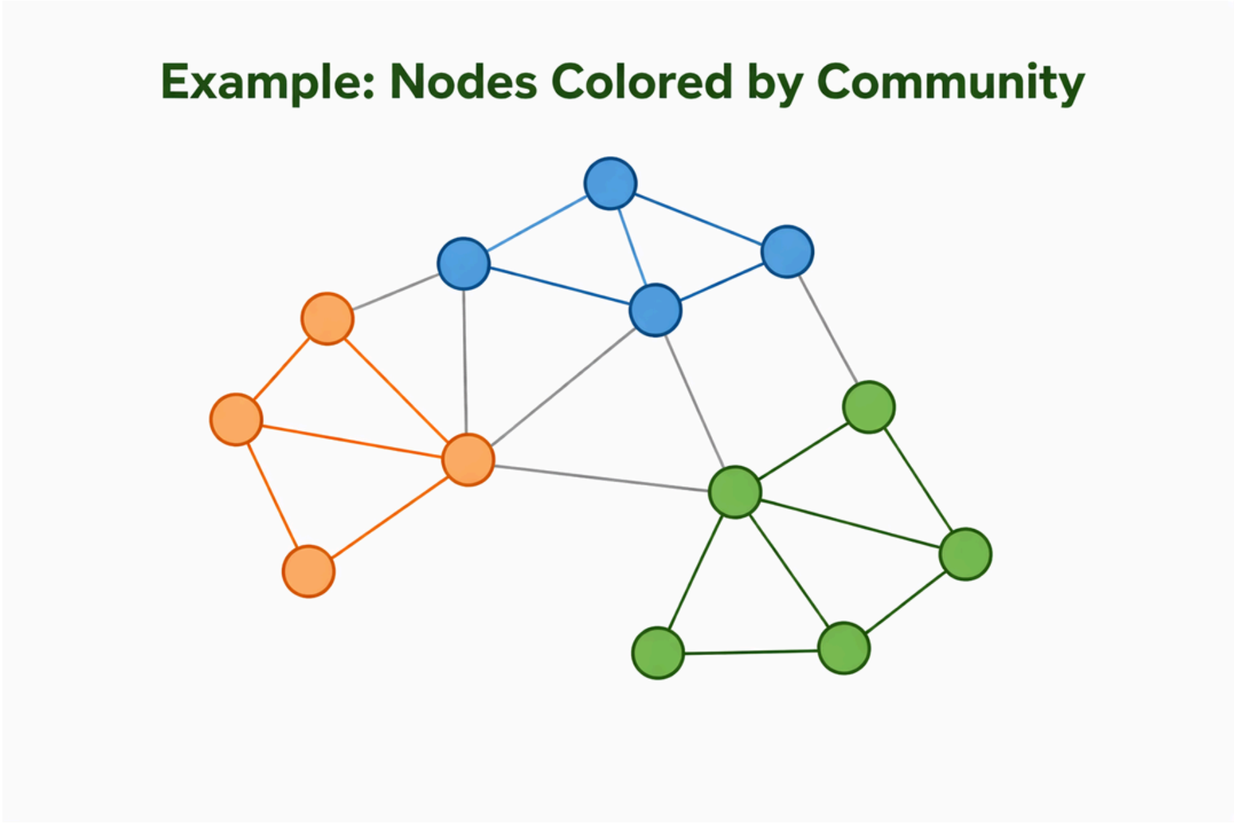
Erna Budhiarti Nababan
Department of Information Technology
Faculty of Computer Science and Information Technology
Universitas Sumatera Utara
Medan, Indonesia
ernabrn@usu.ac.id

Mohammad Andri Budiman
Department of Computer Science, Faculty of Computer Science and Information Technology
Universitas Sumatera Utara
Medan, Indonesia
mandrib@usu.ac.id

Source: [1]]

What is Graph Coloring / Community Coloring?

Graph coloring assigns a color (label) to nodes to make the structure easier to read. In this paper, the workflow is: first run Louvain/Leiden to detect communities, then apply a “community coloring” step where each detected community is given a unique color, so every node is labeled by its community and its community color. The motivation is that community visualizations are hard to interpret without colors, and the authors describe coloring as acting like an “index” that helps the process and interpretation.



Methodology

Leiden Coloring Method:

1) Run Leiden for community detection.
2) Apply graph coloring so each detected community gets a unique color/label (Leiden + coloring = “Leiden Coloring”).
3) Use Degree Centrality inside communities to rank influential nodes (influencers).
4) Compare against Louvain Coloring on modularity, processing time, and number of communities.

Graph Coloring: After communities are detected using Louvain or Leiden, a community coloring step assigns a unique color to each community so every node is labeled with its community and color, making results easier to interpret..

Degree Centrality: Combined with coloring for influencer identification.

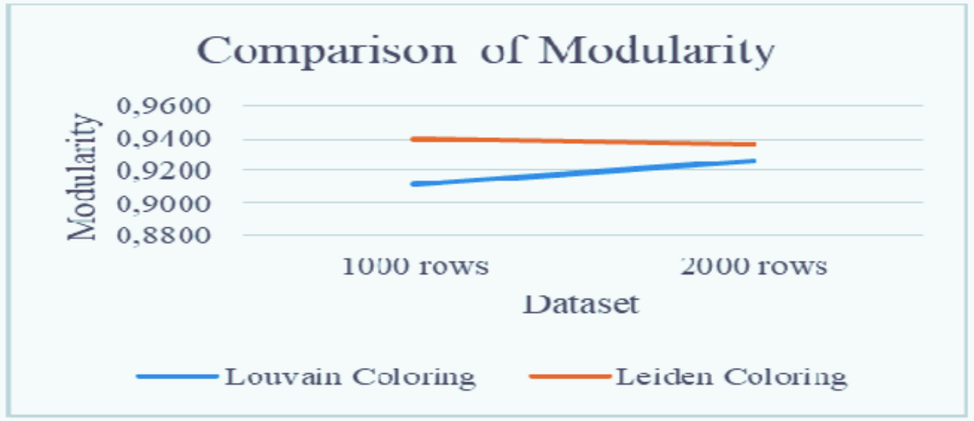


Fig. 4. Comparison of Matrix Modularity

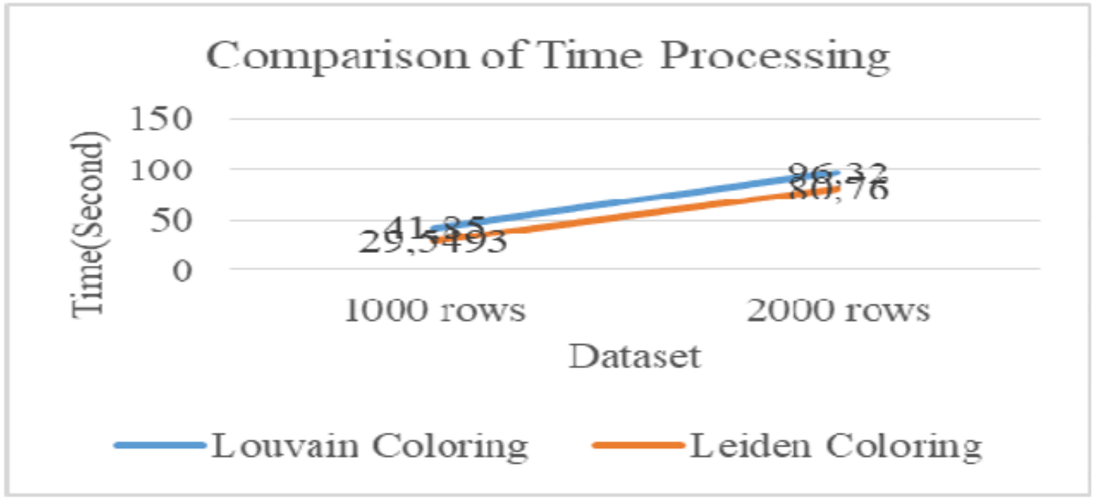


Fig. 5. Comparison of Matrix Processing Time

Source: [2,Fig4,Fig5]]

Results and Key Findings

Key Findings:

- **Modularity**:
  - Louvain: 0.9114 to 0.9259
  - Leiden: 0.9367 to 0.9396
  - Leiden consistently has higher modularity.

- **Processing Time**:
  - Louvain: 69.085 seconds (average)
  - Leiden: 55.154 seconds (average)
  - Leiden is 20% faster.

- **Number of Communities**:
  - Louvain: 1368 communities
  - Leiden: 719 communities
  - Leiden has fewer, more meaningful communities.

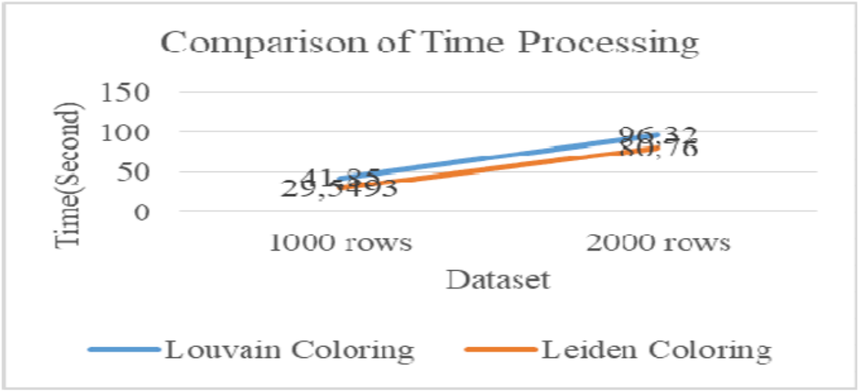


Fig. 5. Comparison of Matrix Processing Time

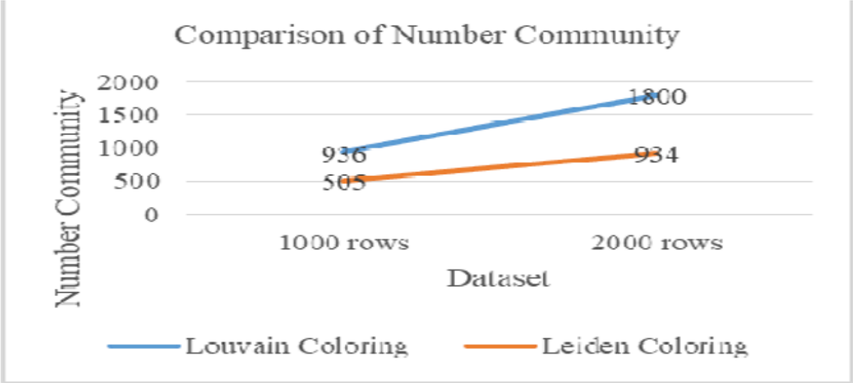


Fig. 6. Comparison of Matrix Number Communities

Dataset	Modularity	
	Louvain Coloring	Leiden Coloring
1000 rows	0.9114	0.9396
2000 rows	0.9259	0.9367
Average	0.9187	0.9383

Source: [2,Fig4,Fig5]]

2025 International Conference on Computer Sciences, Engineering, and Technology Innovation (ICoCSETI)

Conclusion

Key Takeaways:

- Leiden is more efficient than Louvain in influencer detection:
  - Higher modularity,
  - Lower processing time,
  - Fewer communities.

Conclusion:
Using the Twitter (X) keyword dataset “GarudaIndonesia”, Leiden Coloring + Degree Centrality identifies five recommended influencer accounts.
Overall, Leiden Coloring outperforms Louvain Coloring in modularity, processing time, and number of communities.

Community Detection in Large Networks

This study proposes a Weighted Louvain version of community detection, where edges have weights so the algorithm can treat some connections as stronger than others. The goal is to improve community quality (modularity) and check whether weighting can also improve runtime. The authors evaluate the method on several real-world network datasets and compare it against Louvain Base, Leiden, and Girvan–Newman.

2025 11th International Conference on Web Research (ICWR)

Enhancing Community Detection with Weighted Louvain

[1] Akram Karimi Zarandi, [2] Ali Kamandi
[1] School of Engineering Science, College of Engineering, University of Tehran, Iran
karimizarandi@ut.ac.ir
[2] School of Engineering Science, College of Engineering, University of Tehran, Iran
kamandi@ut.ac.ir

Modifications to Louvain Algorithm

The key change is moving from an unweighted graph to a weighted graph, so modularity is optimized using edge weights instead of treating every edge equally. In the experiments, weights are assigned to edges to represent different connection strengths, which makes strong links influence community formation more than weak ones. This helps the algorithm form communities that better reflect dense, meaningful structure while reducing the effect of weaker/noisier links.

Experimental Setup and Datasets

The evaluation uses six real-world networks with different sizes and structures, ranging from small social graphs like Karate Club and Dolphins to larger networks like Power Grid and Hep-th. Performance is measured using modularity (community quality) and execution time (efficiency). The comparisons are reported directly in the paper’s tables and figures across all datasets.

TABLE II. PERFORMANCE **COMPARISON OF COMMUNITY DETECTION ALGORITHMS**

Dataset	*Algorithm*	*Modularity*	*Time (sec)*
Karate Club	Louvain_Base	0.4083	0.0043
	Louvain_Weighted	**0.4336**	**0.0033**
	Girvan-Newman	0.3928	0.0897
	Leiden	0.4198	0.0034
Dolphins	Louvain_Base	0.5224	0.0060
	Louvain_Weighted	**0.5297**	0.0050
	Girvan-Newman	0.5189	0.2613
	Leiden	0.5231	**0.0042**
Football	Louvain_Base	0.6044	0.0109
	Louvain_Weighted	**0.6101**	0.0097
	Girvan-Newman	0.5830	4.9230
	Leiden	0.6046	**0.0053**
Power	Louvain_Base	0.9804	0.2219
	Louvain_Weighted	**0.9898**	0.1588
	Girvan-Newman	0.9879	42.1487
	Leiden	0.9874	**0.0452**
Netscience	Louvain_Base	0.9519	0.1405
	Louvain_Weighted	**0.9598**	0.1314
	Girvan-Newman	0.9504	38.4299
	Leiden	0.9570	**0.1277**
Hep-th	Louvain_Base	0.9534	0.2200
	Louvain_Weighted	**0.9652**	0.1858
	Girvan-Newman	0.9355	145.4406
	Leiden	0.9542	**0.1389**

TABLE I. SUMMARY OF REAL-WORLD NETWORKS USED IN OUR EXPERIMENTS.

Dataset	*Nodes*	*Edges*	*Description*
Zachary's Karate Club	34	78	A small social network representing friendships among members of a university karate club. Commonly used for testing community detection methods.
Dolphin Social Network	62	159	A network of social interactions among dolphins in a marine habitat, based on their observed associations.
American College Football	115	613	A network of games played between college football teams during a single season, where teams are nodes and games form edges.
Power Grid	4,941	6,594	A network illustrating the structure of the US Western power grid, where nodes are power stations and edges represent transmission lines.
Netscience	1,589	2,742	A coauthorship network of researchers working on network science, with connections indicating joint publications.
Hep-th (High-Energy Theory)	8,361	15,751	A collaboration network of scientists in theoretical physics, based on coauthored research papers.

Table I summarizes the datasets, and Table II reports the main results: Weighted Louvain achieves the highest modularity across all datasets. For example, in Karate Club, weighted modularity increases from 0.4083 to 0.4336, and runtime slightly improves from 0.0043s to 0.0033s. Girvan–Newman is consistently far slower, while Leiden is usually fast but not always the top method in modularity.

Source: [2,TABLE I,TABLE II]

Results and Key Findings

Figure 1 shows that Weighted Louvain consistently produces the strongest modularity scores across the tested datasets. Figure 2 shows execution time (log scale): Weighted Louvain is typically faster than Louvain Base and dramatically faster than Girvan–Newman, while Leiden is often the fastest overall. So the results highlight a practical trade-off: weighting improves quality strongly, and speed is competitive but not always the best.

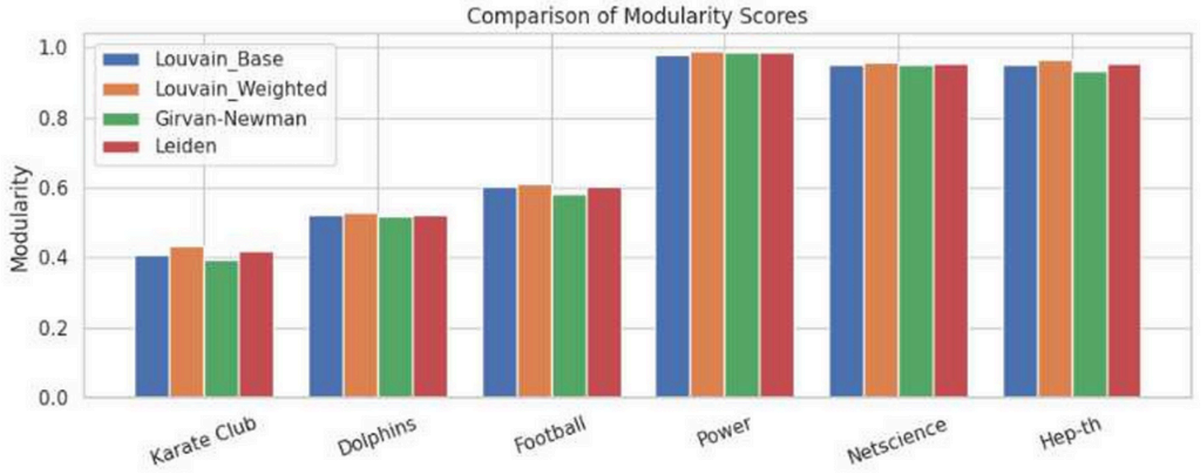


Fig. 1. Comparison of Modularity Scores across different datasets and algorithms.

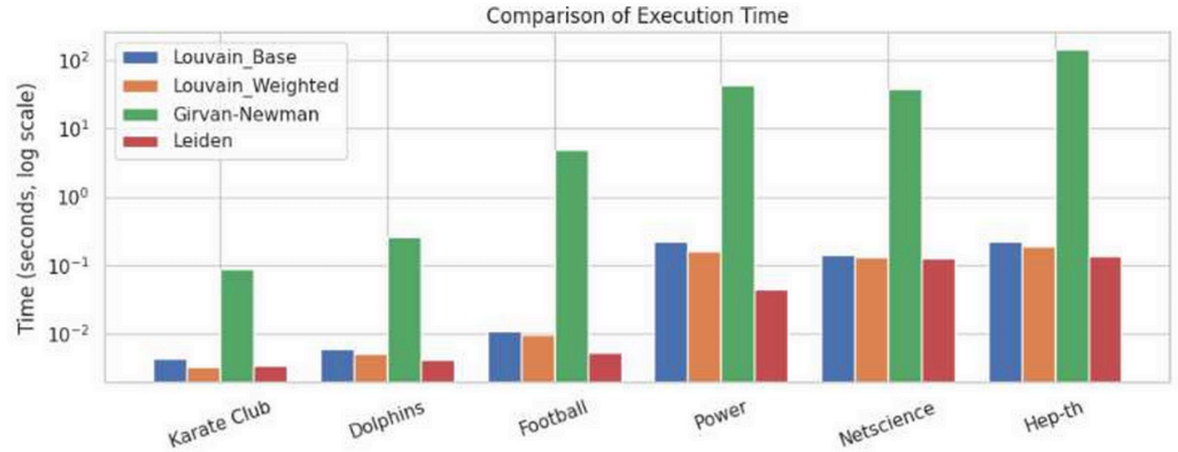


Fig. 2. Comparison of Execution Time for different algorithms.

Source: [2,Fig1,Fig2]]

Modularity and Performance Comparison

This slide highlights the main takeaway from the performance table: Weighted Louvain is the most consistent winner in modularity, but runtime leadership depends on the dataset. In some cases Leiden runs much faster, while Weighted Louvain still gives slightly higher modularity. So the paper's conclusion is not "weighted is always fastest," but rather "weighted gives better communities and remains efficient compared to classic baselines."

Source: [TABLE II]

TABLE II. PERFORMANCE COMPARISON OF COMMUNITY DETECTION ALGORITHMS

Dataset	*Algorithm*	*Modularity*	*Time (sec)*
Karate Club	Louvain_Base	0.4083	0.0043
	Louvain_Weighted	**0.4336**	**0.0033**
	Girvan-Newman	0.3928	0.0897
	Leiden	0.4198	0.0034
Dolphins	Louvain_Base	0.5224	0.0060
	Louvain_Weighted	**0.5297**	0.0050
	Girvan-Newman	0.5189	0.2613
	Leiden	0.5231	**0.0042**
Football	Louvain_Base	0.6044	0.0109
	Louvain_Weighted	**0.6101**	0.0097
	Girvan-Newman	0.5830	4.9230
	Leiden	0.6046	**0.0053**
Power	Louvain_Base	0.9804	0.2219
	Louvain_Weighted	**0.9898**	0.1588
	Girvan-Newman	0.9879	42.1487
	Leiden	0.9874	**0.0452**
Netscience	Louvain_Base	0.9519	0.1405
	Louvain_Weighted	**0.9598**	0.1314
	Girvan-Newman	0.9504	38.4299
	Leiden	0.9570	**0.1277**
Hep-th	Louvain_Base	0.9534	0.2200
	Louvain_Weighted	**0.9652**	0.1858
	Girvan-Newman	0.9355	145.4406
	Leiden	0.9542	**0.1389**

computational costs, making it impractical for large networks.

Conclusion

Overall, the paper shows that adding weights can consistently improve modularity and often reduces runtime compared to the original Louvain and especially Girvan–Newman. The results also suggest that algorithm choice depends on the goal: Weighted Louvain for higher-quality communities, and Leiden when runtime is the top priority. Future work would be stronger if weights came from real relationship strength (instead of assigned weights) and if the approach were tested on even larger graphs or parallel implementations.

Lessons Learned & conclusion

1- We learned how to read research papers efficiently by focusing on the problem, the core method, and the evaluation/results section.
2- We learned that “better community detection” depends on the goal: higher modularity, faster runtime, or fewer/more communities can each matter depending on the application.
3- We learned that scalability is the main real-world challenge, so techniques like distributed processing or algorithm refinements are essential for large graphs.
4- We learned that combining community detection with extra ideas (like edge weights or influencer metrics such as centrality) makes the method more useful for real applications.

CITATIONS

Distributed Community Detection in Web-Scale Networks

[1]M. Ovelgonne, "Distributed Community Detection in Web-Scale Networks," Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), 2013, pp. 66-70. [Online]. Available: https://ieeexplore.ieee.org/document/6785689. [Accessed: Dec. 10, 2025].

Enhancing Community Detection with Weighted Louvain

[2]A. Karimi Zarandi and A. Kamandi, "Enhancing Community Detection with Weighted Louvain," 2025 11th International Conference on Web Research (ICWR), 2025, pp. 579-583. [Online]. Available: https://ieeexplore.ieee.org/document/11006186. [Accessed: Dec. 10, 2025].

A Comparative Analysis of Louvain and Leiden Coloring Algorithms in Influencer Detection

[3]H. Handrizal, M. A. Budiman, P. Sihombing, and E. B. Nababan, "A Comparative Analysis of Louvain and Leiden Coloring Algorithms in Influencer Detection," 2025 International Conference on Computer Sciences, Engineering, and Technology Innovation (ICoCSETI), 2025, pp. 594-600. [Online]. Available: https://ieeexplore.ieee.org/document/11070425. [Accessed: Dec. 10, 2025].

Thank You!

For your listening...

Elaborate on what you want to discuss.