Parameter-Free Community Detection via Gomory-Hu Trees and Modularity

Deokhyeon Kim (20201032)
UNIST
South Korea
ejrgus1404@unist.ac.kr

Abstract

We propose a recursive graph clustering algorithm that combines global min-cut structures with modularity-based evaluation. By leveraging Gomory-Hu trees, the algorithm explores all-pairs minimum cuts to identify sparse connectivity regions, and recursively partitions the graph based on balanced cuts. Each subgraph is evaluated using a modularity threshold that scales with component size, allowing automatic termination without manual parameters. To improve the quality of detected communities, the algorithm discards singleton components (dangling nodes) that tend to degrade modularity. We implement the method in C++ and evaluate it on four benchmark datasets: Karate Club, Dolphin, Football, and Polblogs. Our results show that the algorithm effectively captures community structure in small or well-separated networks, though it may underperform on large or complex graphs. These findings highlight the promise of flow-based cut structures as a foundation for interpretable and parameter-free graph clustering.

1 INTRODUCTION

Clustering is a fundamental task in unsupervised learning that aims to group similar data points into meaningful subsets. In graph-based settings, clustering often translates into community detection—identifying tightly connected groups of nodes within a larger network. Applications range from social network analysis and biological data mining to recommendation systems and fraud detection.

While classical clustering algorithms such as K-means or DBSCAN perform well on vectorized data, they are not directly applicable to general graphs due to their reliance on geometric distance metrics. In contrast, graph-specific methods such as spectral clustering or the Louvain algorithm are designed to identify communities by optimizing modularity—a quality function that quantifies how well a network is divided into densely connected groups.

Despite their success, many existing approaches suffer from limitations such as reliance on local heuristics, inability to guarantee globally optimal cuts, or sensitivity to initialization and resolution parameters.

In this paper, we propose a novel graph clustering algorithm that combines Gomory-Hu trees and modularity-based evaluation. Our method recursively partitions the graph using global minimum cuts derived from the Gomory-Hu tree and evaluates each resulting subgraph based on a modularity threshold. This hybrid approach enables globally-aware graph partitioning while maintaining flexibility through a data-dependent stopping condition.

We evaluate our method on four benchmark datasets and compare its performance to known ground truth labels in terms of modularity and the number of detected communities. Although the approach performs best on simpler networks, it highlights the potential of using exact flow-based cut structures for interpretable and principled graph clustering.

2 RELATED WORK

Graph clustering is a diverse field encompassing a variety of algorithmic paradigms. These include spectral methods, statistical inference models, neural embedding techniques, modularity optimization, and flow-based partitioning. In this work, we focus on the intersection of modularity-based heuristics and cut-based partitioning, which form the conceptual backbone of our proposed approach.

Modularity-Based Clustering

Modularity is one of the most widely used quality functions for community detection. It measures the difference between the observed intra-cluster edge density and that expected in a random graph. Algorithms such as Louvain [1] and Leiden [11] optimize modularity using greedy, multi-level strategies. While effective and scalable, these approaches can suffer from issues such as resolution limits [3] and sensitivity to initialization or local minima.

Cut-Based and Flow-Based Partitioning

Cut-based methods aim to divide graphs by minimizing the number or total weight of edges between parts. Classical min-cut algorithms [10] and multilevel partitioning frameworks like METIS [5] are widely used in applications requiring balanced partitioning. The Gomory-Hu Tree [4] extends the notion of minimum cuts to all vertex pairs by constructing a tree with $n - 1$ maximum flow computations. Despite its compactness and global view of connectivity, its use in clustering remains rare.

Comparison and Contribution

Our method integrates Gomory-Hu trees with modularity evaluation in a recursive graph clustering algorithm. Unlike Louvain or Leiden, which rely on greedy local updates, we explore global cut structures to guide partitioning. In contrast to traditional cut-based methods, we evaluate clusters based on modularity, not balance or volume.

Importantly, our algorithm is fully automatic: it requires no user-defined parameters beyond the input graph. The modularity threshold is internally computed as a function of subgraph size, making the method simple, reproducible, and robust to overfitting or poor parameter choice.

To the best of our knowledge, our work is the first to combine Gomory-Hu tree-based recursive partitioning with modularity-aware stopping criteria for general-purpose graph clustering.

3 PROBLEM STATEMENT

Given an undirected, unit capacity graph $G = (V, E)$, the goal of graph clustering is to partition the vertex set V into disjoint subsets $C_1, C_2, \ldots, C_k$ such that each subset corresponds to a meaningful community in the network. Ideally, nodes within the same cluster are densely connected, while nodes across different clusters are sparsely connected.

In this work, we focus on identifying such communities using a recursive graph partitioning approach guided by both global connectivity and community quality metrics. Specifically, we aim to:

- Detect non-trivial substructures in G by computing minimum cuts using Gomory-Hu Trees, which capture all-pairs minimum cuts compactly.
- Use modularity as the objective function to determine whether a subgraph forms a sufficiently cohesive community.
- Automatically determine the number and size of clusters without any manual tuning or user-defined parameters.

Unlike existing methods that rely on local greedy heuristics or fixed-size balanced partitioning, our approach explores global cut structures and employs a data-dependent stopping condition based on modularity. The problem can thus be summarized as follows:

Input: An undirected, unit capacity graph $G = (V, E)$.
Output: A set of clusters $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ such that $\bigcup C_i \subseteq V$ and each C_i satisfies a modularity threshold.

4 ALGORITHM

We propose a recursive graph clustering algorithm that leverages Gomory-Hu trees to guide the division of graph components, using modularity as a stopping criterion. Our approach identifies high-quality communities while avoiding trivial or noisy structures.

4.1 Overview

The algorithm partitions the input graph into clusters through the following steps:

(1) **Component Extraction:** Decompose the graph into connected components.
(2) **Modularity Evaluation:** For each component, compute its modularity. If it exceeds a size-dependent threshold, the component is accepted as a cluster.
(3) **Gomory-Hu Tree Construction:** Otherwise, construct the Gomory-Hu tree of the component to determine all-pairs min-cuts.
(4) **Optimal Cut Selection:** Among the min-cuts of the component, select the one that *maximizes the size of the smaller partition* after the cut. This prioritizes balanced splits over highly imbalanced ones.
(5) **Component Splitting:** Partition the component using the selected cut and recursively apply the algorithm to each subcomponent.
(6) **Dangling Node Removal:** Components of size 1 are discarded and excluded from clustering.

Algorithm 1 Clustering

```
function ClusterGraph(G)
    C ← ∅                                              ▷ Cluster set
    PQ ← empty priority queue
    Components ← connected components of G
    for all comp ∈ Components do
        if |comp| = 1 then
            continue
        else if Modularity(comp) ≥ Threshold(|comp|) then
            C ← C ∪ {comp}
        else
            compute_optcut(comp)
            PQ.push(comp)
        end if
    end for
    while PQ is not empty do
        comp ← PQ.pop()
        (comp_1, comp_2) ← Split(comp)
        for all c ∈ {comp_1, comp_2} do
            if |c| = 1 then
                continue
            else if Modularity(c) ≥ Threshold(|c|) then
                C ← C ∪ {c}
            else
                compute_optcut(c)
                PQ.push(c)
            end if
        end for
    end while
    return C
end function
```

4.2 Modularity Threshold

We define the modularity threshold as a function of component size:

$$T(s) = 0.025 \cdot \sqrt{s}$$

This threshold function is based on the observation that:

- **Small clusters** tend to have inherently low modularity due to few internal edges and relatively large boundary.
- **Large clusters** are more likely to exhibit high modularity due to their scale, even with moderate density.

Thus, a sublinear threshold $T(s) \propto \sqrt{s}$ is used to allow more lenient criteria for smaller clusters and stricter ones for larger clusters. This balances the risk of rejecting meaningful small communities and accepting spurious large ones. The coefficient 0.025 was determined empirically and held constant across all experiments.

4.3 Time Complexity

The total time complexity is effectively governed by the cost of constructing Gomory-Hu trees. Other steps in the algorithm, such as modularity calculation and connected component detection, run in linear or near-linear time with respect to the number of nodes and edges.

Each Gomory-Hu tree for a component with n nodes and m edges requires $n-1$ max-flow computations. Using Dinic's algorithm, each max-flow runs in:

$$O\left(\min\left(n^{2/3}\cdot m,\ m^{3/2}\right)\right)$$

This complexity applies specifically to unit-capacity graphs, where Dinic performs more efficiently due to structural properties of the residual graph.

Therefore, building a Gomory-Hu tree takes:

$$O\left(n\cdot\min\left(n^{2/3}\cdot m,\ m^{3/2}\right)\right)$$

Let N and M be the number of nodes and edges in the full input graph. If the algorithm makes $O(N)$ recursive calls in the worst case, then the total time complexity becomes:

$$O\left(N^2\cdot\min\left(N^{2/3}\cdot M,\ M^{3/2}\right)\right)$$

This bound assumes that each recursive call operates on a component with $O(N)$ nodes and $O(M)$ edges. In practice, however, early termination on high-modularity components significantly reduces the number and size of recursive calls, making the worst-case complexity rarely observed.

5 EXPERIMENTS

We evaluated our proposed clustering algorithm on four widely used real-world network datasets: *Karate Club*, *Dolphin*, *Football*, and *Polblogs*. All experiments were conducted using a C++ implementation.[1]

Datasets

- **Karate Club:** A social network of 34 nodes representing friendships among members of a karate club.
- **Dolphin:** A network of 62 dolphins with observed community structure.
- **Football:** A graph of 115 college football teams connected by games played, with 12 known conferences.
- **Polblogs:** A network of 1,224 political blogs labeled as liberal or conservative. Although the dataset is originally a directed graph, we treated it as undirected by assuming an undirected edge exists if any directional edge is present between two nodes.

Each dataset was provided in edge list format, with community labels serving as ground truth. All edges were treated as undirected and unit capacity for consistency across datasets.

Evaluation Metric

We use the modularity score as the primary evaluation metric. Modularity quantifies the extent to which a given clustering exhibits dense intra-cluster connectivity and sparse inter-cluster connections. We also compare the number of detected clusters against the known ground truth.

[1]https://github.com/dh28be/CSE304_Project

Dataset	# GT Clusters	# Exp Clusters	GT Modularity	Exp Modularity
Karate	2	0	0.3715	0.0000
Dolphin	2	2	0.3735	0.2490
Football	12	1	0.5540	0.1922
Polblogs	2	1	0.4055	0.9804

Table 1: Comparison of Ground Truth and Experimental Results

Case Study: Dolphin Dataset

To better illustrate our algorithm's performance, we visualize the Dolphin network with both ground truth labels and our clustering results. As shown in Figure 1, the left plot displays the ground truth community structure, while the right plot shows the output of our method. Discarded nodes are colored in gray.

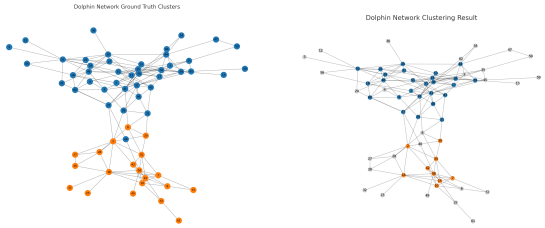

Figure 1: Dolphin network community structure: Ground truth (left) vs. experimental result (right)

Analysis

The algorithm shows strong performance on the Dolphin dataset, successfully identifying two communities with a modularity score of 0.2490—relatively close to the ground truth of 0.3735. This suggests the method is effective in small- to mid-sized graphs with well-separated community structure.

In the Karate dataset, no clusters were detected. This is likely due to the modularity threshold function rejecting all components, which indicates that the threshold may be overly strict for small graphs.

For the Football and Polblogs datasets, only one cluster was identified, despite multiple ground truth communities. In the case of Polblogs, this single detected cluster achieved an unusually high modularity score (0.9804), which suggests that the entire network was interpreted as one cohesive community. This can occur in large graphs when the algorithm fails to find sparse enough cuts under the given threshold.

Overall, while the proposed algorithm is capable of extracting meaningful community structures in small or cleanly separated graphs, it is currently limited in its ability to resolve more complex or large-scale community structures. Future improvements could include adaptive thresholding or integration with post-processing refinement techniques.

6 CONCLUSION

We introduced a novel graph clustering algorithm that combines global cut information from Gomory-Hu trees with modularity-based validation. Unlike many existing methods that rely on local heuristics or iterative refinements, our approach employs exact minimum cuts to guide recursive partitioning and uses a size-adaptive modularity threshold as a principled stopping condition.

Through experiments on several benchmark datasets—*Karate Club*, *Dolphin*, *Football*, and *Polblogs*—we demonstrated that the algorithm is effective in extracting meaningful community structures in small and well-separated networks. In particular, the Dolphin dataset was successfully partitioned into two high-quality clusters.

However, the method shows limitations on more complex or large-scale networks such as *Football* and *Polblogs*. In these cases, it often returns a single cluster, as the static modularity threshold fails to detect finer-grained structures. This highlights the challenge of using fixed thresholds and recursive min-cuts in densely interconnected networks.

Future work will explore several directions to improve our approach. One key enhancement is to make the modularity threshold adaptive to graph characteristics, allowing more flexibility across diverse network types. Additionally, incorporating local refinement techniques—such as modularity optimization within detected clusters—may help capture finer community structures. We also plan to extend the algorithm to handle directed and weighted graphs. Ultimately, our goal is to further enhance the scalability and robustness of our method while preserving its core advantage: fully *parameter-free* community detection grounded in flow-based global cut structures.

References

[1] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* 2008, 10 (2008), P10008. doi:10.1088/1742-5468/2008/10/P10008

[2] Chandra Chekuri and David Morrison. 2010. Lecture 6: Gomory-Hu Trees. https://courses.grainger.illinois.edu/cs598csc/sp2010/Lectures/Lecture6.pdf. Accessed: 2025-05-27.

[3] Santo Fortunato and Marc Barthélemy. 2007. Resolution limit in community detection. *Proceedings of the National Academy of Sciences* 104, 1 (2007), 36–41. doi:10.1073/pnas.0605965104

[4] Ralph E Gomory and T. C. Hu. 1961. Multi-terminal network flows. *J. Soc. Indust. Appl. Math.* 9, 4 (1961), 551–570. doi:10.1137/0109045

[5] George Karypis and Vipin Kumar. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. In *Proceedings of the 1998 International Conference on Parallel Processing (ICPP)*. IEEE, 113–122. doi:10.1109/ICPP.1998.709625

[6] koosaga. 2016. Flow Algorithms Summary. https://koosaga.com/18. Accessed: 2025-05-27.

[7] mons1220. 2022. Network Modularity. https://mons1220.tistory.com/93. Accessed: 2025-05-27.

[8] Koutris Paraschos and Vasileios Syrgkanis. 2008. Gomory-Hu Trees: Theory and Applications. https://corelab.ntua.gr/seminar/material/2008-2009/2008.10.20.Gomory-Hu-trees-and-applications.slides.pdf. Accessed: 2025-05-27.

[9] ShahjalalShohag. 2024. Gomory Hu Tree.cpp. https://github.com/ShahjalalShohag/code-library/blob/main/Graph-Theory/Gomory-Hu-Tree.cpp. Accessed: 2025-05-27.

[10] Mechthild Stoer and Frank Wagner. 1997. A simple min-cut algorithm. *Journal of the ACM (JACM)* 44, 4 (1997), 585–591. doi:10.1145/263867.263872

[11] Vincent A Traag, Ludo Waltman, and Nees Jan van Eck. 2019. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports* 9, 1 (2019), 5233. doi:10.1038/s41598-019-41695-z