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

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

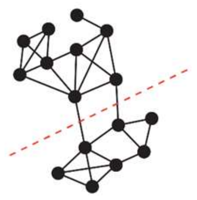
Introduction

Community detection plays a crucial role in understanding the structural organization of complex networks, such as social interactions, citation relationships, and biological systems. Traditional clustering techniques, like K-means or DBSCAN, are not directly applicable to general graphs due to their reliance on geometric assumptions and fixed parameters. Even graph-specific methods such as the Louvain algorithm often suffer from heuristic limitations and parameter sensitivity, making their results difficult to interpret or reproduce. This motivates the need for a graph clustering method that is both principled and parameter-free, enabling robust and interpretable community detection without manual tuning. Our approach addresses this gap by leveraging Gomory-Hu trees and modularity-based validation to detect high-quality communities in a recursive and automated manner.

Background Knowledge

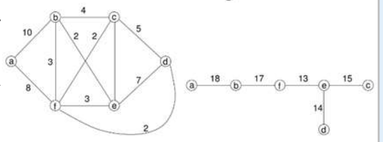
Min-Cut

The smallest set of edges whose removal disconnects a graph into two parts.



Gomory-Hu Tree

A tree structure that compactly represents all-pairs min-cuts in an undirected graph using only $n-1$ max-flow computations.



Modularity

A score that measures the quality of a clustering by comparing the density of edges within communities to that of a random graph.

$$Q = \frac{1}{2M}\sum_{i,\,j}^{N}\left(a_{ij} - \frac{k_i k_j}{2M}\right)\delta[C(i),\ C(j)]$$

Related Work

Among various graph clustering techniques, two lines of work are most relevant to our approach: modularity-based methods and cut-based partitioning.

Modularity optimization algorithms, such as Louvain and Leiden, aim to maximize a quality score that favors dense intra-community connections. These methods are widely used due to their scalability, but they rely on greedy heuristics and often suffer from resolution limits.

Cut-based methods, including METIS and spectral clustering, focus on minimizing edge cuts to divide the graph. While effective in balanced partitioning, they typically require user-defined parameters and do not explicitly consider community quality.

Our method combines both perspectives: we use Gomory-Hu trees to extract global min-cuts, and apply modularity threshold to determine meaningful clusters. This hybrid approach is parameter-free, globally informed, and directly aligns with the goal of community detection.

Problem Statement

Input

An undirected, unweighted graph $G = (V, E)$

Goal

Partition the vertex set V into clusters $\{C_1, C_2, \ldots, C_k\}$ such that:

- Nodes within the same cluster are densely connected
- Nodes in different clusters are sparsely connected
- Each cluster satisfies an adaptive modularity threshold

Approach

Use Gomory-Hu trees to find global min-cuts and recursively divide the graph.
A cluster is accepted if its modularity exceeds a size-dependent threshold.
No user-defined parameters are required.

Algorithm

We propose a recursive graph clustering algorithm that combines global min-cut structures with modularity-based validation.

Algorithm

1. **Component Extraction**
   Identify connected components of the input graph.
2. **Modularity Check**
   Accept components as clusters
   if their modularity exceeds size-dependent threshold.
3. **Gomory-Hu Tree Construction**
   For components that fail the modularity check,
   compute all-pairs min-cuts using a Gomory-Hu tree.
4. **Balanced Cut Selection**
   Among the min-cuts choose the one that
   maximizes the size of the smaller partition after the cut.
5. **Recursive Splitting**
   Split the component using the selected cut,
   and repeat the process for each subcomponent.
6. **Singleton Removal**
   Discard any isolated single-node components.

This approach is fully parameter-free, scalable, and interpretable.

Modularity Threshold

To determine whether a subgraph forms a valid community, we apply a size-dependent threshold defined as:

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

where s is the number of nodes in the component.

The use of $\sqrt{s}$ reflects the fact that small components tend to have lower modularity due to their limited internal connectivity, whereas large components are more likely to have higher modularity even with moderate cohesiveness.

The coefficient 0.025 was chosen empirically based on experiments across benchmark datasets and kept fixed throughout all evaluations. It enables consistent, parameter-free clustering behavior across graphs of different sizes.

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

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

Time Complexity

The total time complexity is dominated by Gomory-Hu tree construction.
Other steps, such as modularity evaluation and component detection, run in linear or near-linear time.

For a component with n nodes and m edges, building a Gomory-Hu tree requires $n-1$ max-flow computations.
Using Dinic's algorithm on unit-capacity graphs, each max-flow runs in:

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

Thus, a single Gomory-Hu tree construction takes:

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

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

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

In practice, early termination based on modularity threshold significantly reduces the depth and size of recursive calls, making the algorithm efficient for real-world graphs.

Experiments

We evaluated our algorithm on four real-world datasets: Karate Club, Dolphin, Football, and Polblogs.
Each dataset varies in size and structure, testing the generality and robustness of our approach.

All experiments used a C++ implementation. Edges were treated as undirected and unweighted, and modularity was used as the evaluation metric.

	# GT Cls	# Exp Cls	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

Table1. Comparison of Ground Truth and Experimental Results

The results show that our method performs well on small or well-separated networks. For example, in the Dolphin graph, the algorithm successfully detected two communities with a modularity of 0.2490, close to the ground truth of 0.3735.

However, on more complex graphs like Football and Polblogs, the algorithm tended to return a single cluster. In the Polblogs case, this resulted in a very high modularity (0.9804), suggesting that the entire graph was interpreted as a single cohesive structure.

In contrast, no clusters were detected in the Karate Club graph, likely due to our modularity threshold rejecting all components as insufficiently cohesive.

These findings indicate that while the algorithm is effective for small, structured graphs, it can struggle with large or densely connected networks—highlighting the need for improvements on adaptive threshold or post-processing in future work.

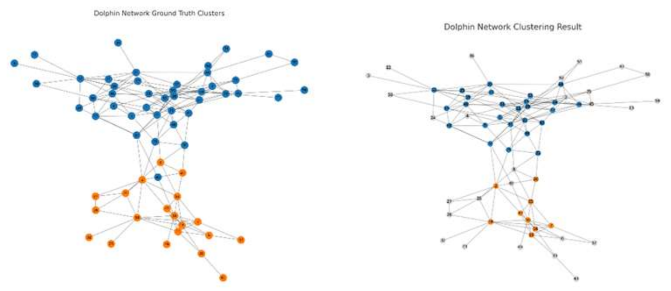

Dolphin Network Ground Truth Clusters
Dolphin Network Clustering Result

Fig1. Dolphin Network Community Structure:
Ground Truth (left) vs. Experimental Result (right)

Conclusion

Contribution

We propose a recursive graph clustering algorithm that uses Gomory-Hu Trees to find global min-cuts, combined with a modularity-based validation.
The method is fully parameter-free and interpretable, providing a simple and consistent approach to graph clustering.

Strengths

Our approach detects meaningful communities in small and well-separated networks, without relying on manual tuning.
The use of structural cuts and modularity threshold leads to consistent and explainable results.

Limitations

In dense or complex graphs, the algorithm may fail to find meaningful partitions and return a single cluster.
This suggests that the algorithm is cautious about forming communities and may overlook subtle structures when the modularity score is just below the threshold.

Future Work

We aim to improve the algorithm's flexibility and accuracy in several ways.

First, making the modularity threshold adaptive to graph size or density could enhance performance across diverse networks.

We also plan to apply local refinement techniques—such as modularity optimization within clusters—to detect finer structures.

For singleton nodes, we propose a post-processing step that traces their cut history and reassigns them to appropriate clusters.

Lastly, extending the method to directed and weighted graphs will broaden its applicability, while maintaining its parameter-free, interpretable nature.