2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining

Constant Community Identification in Million Scale Networks Using Image Thresholding Algorithms

Anjan Chowdhury
Center for Soft Computing Research (CSCR)
Indian Statistical Institute
Kolkata, India
anjan_r@isical.ac.in

Sriram Srinivasan
Department of Radiation Oncology
National Radiation Oncology Program
Virginia Commonwealth University
Veterans Healthcare Administration, VA, USA
sriram.srinivasan@vcuhealth.org

Sanjukta Bhowmick
Department of Computer Science
University of North Texas
North Texas, USA
sanjukta.bhowmick@unt.edu

Animesh Mukherjee
Department of Computer Science and Engineering
IIT Kharagpur
Kharagpur, India
animeshm@cse.iitkgp.ac.in

Kuntal Ghosh
Machine Intelligence Unit and CSCR
Indian Statistical Institute
Kolkata, India
kuntal@isical.ac.in

***Abstract*—Constant communities, i.e., groups of vertices that are always clustered together, independent of the community detection algorithm used, are necessary for reducing the inherent stochasticity of community detection results. Current methods for identifying constant communities require multiple runs of community detection algorithm(s). This process is extremely time consuming and not scalable to large networks. We propose a *novel approach for finding the constant communities, by transforming the problem to a binary classification of edges*. We apply the Otsu method from image thresholding to classify edges based on whether they are always within a community or not. *Our algorithm does not require any explicit detection of communities* and can thus scale to very large networks of the order of millions of vertices. Our results on real-world graphs show that our method is significantly faster and the constant communities produced have higher accuracy (as per F1 and NMI scores) than state-of-the-art baseline methods.**

***Index Terms*—Constant communities, Otsu method, multi-Otsu**

I. Introduction

Community detection, i.e., finding clusters of tightly connected vertices, is a fundamental problem in network analysis with diverse applications from finding genes with similar functions to tracking terrorist cells via social networks.

Stochasticity in community detection. Community detection algorithms are typically based on optimizing parameters, such as modularity or entropy and are NP-complete. Clusters obtained through community detection can differ based on the methods used, choice of parameters, and even the order of the vertices. It is difficult to separate the algorithm artifacts from the information about the network structure. A critical problem in community detection is how to reduce the stochasticity and produce reliable results.

Constant communities. Stable clusters can be obtained by identifying *constant communities* [1]. Constant (or consensus) communities are groups of vertices that are always clustered together for all non-trivial community detection algorithms, and thus form the *stable and invariant* subset of the possible reasonable community detection results.

Current methods for finding constant communities require the results from multiple community detection algorithms (or the same algorithm with different parameters). This process is expensive in terms of time and memory and consequently, current methods do not scale to large networks.

We address this challenge by developing efficient algorithms for finding constant communities that are based on the binary classification of edges and can scale to large networks with an order of millions of vertices.

Our contribution and key ideas. We present a scalable and efficient method for identifying constant communities by transforming this problem into a binary classification task. Our method is based solely on the local structural properties of the edges and, requires no communities as input. Our algorithm is based on these **two key ideas**;

First, for each edge, we obtain a set of four features based on how well the neighbors of the endpoints are connected with each other. Using these features, we identify how likely the edge will be a within-community edge.

Second, we apply variations of Otsu's algorithm [2], a method for thresholding images, to obtain a binary classification of the edges based on these features. Otsu's algorithm allows for completely unsupervised classification, thus no information about community structure is needed a-priori. This reduces the execution time of our algorithm drastically.

ASONAM '21, November 8-11, 2021, Virtual Event, Netherlands

ACM ISBN 978-1-4503-9128-3/21/11?/$15.00
http://dx.doi.org/10.1145/3487351.3488350

Evaluation. We use **three metrics** to evaluate our results. *First*, using F1-score we show that our methods outperform the most competitive baselines. *Second*, we reconstruct the constant communities based on the classified edges and compare with the ground truth communities. Our method obtains higher or comparable NMI (Normalized Mutual Information) scores to the other baseline methods. *Third*, we report the execution time. We are 10 times faster than the baseline algorithms for the smaller networks. On large networks, our methods finish in a few hours to a day, while the baseline methods did not finish even after running for several days.

II. Overview of Otsu's Algorithm

Binary Otsu. Otsu's binary classification [2] is used to binarize grayscale images such that the object in the foreground can be distinguished from the background. The output is a single intensity threshold that separates pixels into two classes: foreground and background. To determine this optimal threshold, the threshold values are adjusted to maximize the variance in intensity between the two classes.

Multi-level Otsu. As shown in Figure 1, the single threshold based binarization may not always yield the best results. In this case, Otsu's binary-level thresholding [2] can be extended to multi-level thresholding [3], to determine multiple thresholds to segment an image into clusters and recognize the various parts. We have used the Two-Stage Multi-threshold Otsu algorithm (TSMO) [3].

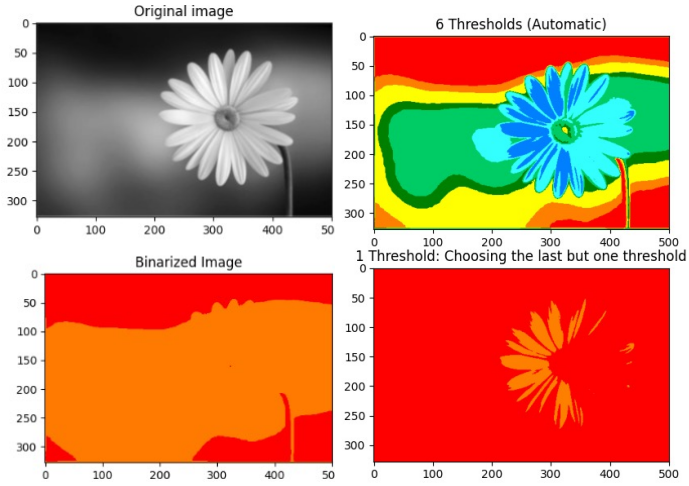

Fig. 1: Otsu's binarization and multi-level techniques to obtain a threshold that delineates the foreground of an image from its background. Binarization (bottom left) may not always give the desired result. Here the flower (object) has not been delineated from its background. Multiple thresholds may be required for an improved object-background delineation (right side figures).

III. Applying Otsu's Method for Finding Constant Communities

We describe how we applied Binary and TSMO (henceforth called **Multi-Otsu**) for finding constant communities, using the following steps. A schematic diagram is given in Figure 2.

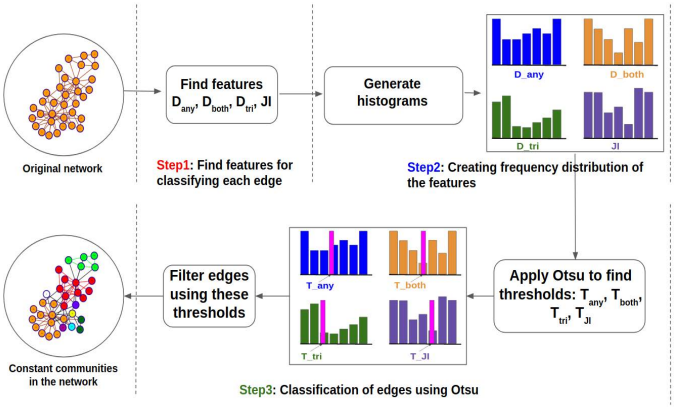

Fig. 2: The steps for finding the constant communities.

A. Step 1: find features for classifying each edge

The likelihood that an edge (u, v) will always be within a community is determined by the connections between the neighbors of the endpoints. Let $N(v)$ be the set of neighbors of vertex v; $\Delta(v)$ be the set of triangles that contain vertex v; $\Gamma(X)$, be the density of the subgraph induced by the vertices in set X. We classify an edge (u, v), based on these features;

- Density of the subgraph induced by neighbors of *both* u *and* v; $D_{both}(u, v) = \Gamma(N(v) \cap N(u))$.
- Density of the subgraph induced by u and v and their neighbors; $D_{any}(u, v) = \Gamma(N(v) \cup N(u))$.
- Ratio of the triangles containing both endpoints to those containing at least one; $D_{tri}(u, v) = \frac{|\Delta(u) \cap \Delta(v)|}{|\Delta(u) \cup \Delta(v)|}$.
- Jaccard index of the neighbors; $JI(u, v) = \frac{|N(v) \cap N(u)|}{|N(v) \cup N(u)|}$.

B. Step 2: create frequency distribution of features.

Let **B100** be the set of edges that are always within a community, and **B0** be the remaining set of edges, that may or may not be within communities. An edge is classified into **B100** if any of the following conditions hold.

(i) the percentage of common neighbors is high (i.e., high JI) *and also* the density of the subgraph induced by them is high (i.e., high D_{both}). The edge has many common neighbors, and the neighbors and the edge form a dense subgraph.

(ii) the density of the subgraph induced by all the neighbors of u and v is high (i.e., high D_{any}).

(iii) the ratio of the number of triangles containing both endpoints, to the number of triangles with at least one endpoint is high (i.e., high D_{tri}). The edge supports many triangles, thus its endpoints are likely to be co-clustered.

C. Step 3: classification of edges using Otsu

We now use Otsu to find a threshold to classify the values of edges as high and low.

Classification using binary Otsu (Algorithm 1). We find the optimal threshold for each feature, and then classify the edges as per the rules in Section III-B. A feature has high value if the value is higher than the computed threshold.

Algorithm 1: Edge classification using binary Otsu method.

Input : The network $G(V, E)$. Feature sets $D_{both}, D_{any}, D_{tri}$ and JI consisting features for all the edges.
Output: Edges classified into **B100** (in constant community) or **B0** (not in constant community).

```
/* Generate histograms and compute optimal threshold
   via Otsu's method                                   */
[T_{D_both}] ← OtsuMethod(D_both)
[T_{D_any}] ← OtsuMethod(D_any)
[T_{D_tri}] ← OtsuMethod(D_tri)
[T_JI] ← OtsuMethod(JI)
/* Classify the edges                                  */
B100 ← Φ; B0 ← Φ;
forall e ∈ E do
    if ((D_both(e) > T_{D_both}/2 AND JI(e) > T_JI/2) OR
    (D_any(e) > T_{D_any} OR D_tri(e) > T_{D_tri}/2)) then
        B100 ← B100 ∪ e
    else
        B0 ← B0 ∪ e
```

Algorithm 2: Edge classification using multi-Otsu.

Input : The network $G(V, E)$, Feature sets $D_{both}, D_{any}, D_{tri}$ and JI consisting features for all the edges.
Output: Edges classified into **B100** (constant community) or **B0** (not constant community).

```
/* Generate histograms and compute multiple thresholds
   list via Multi-Otsu's method                        */
T_{both_set} ← MultiOtsuMethod(D_both)
T_{any_set} ← MultiOtsuMethod(D_any)
T_{tri_set} ← MultiOtsuMethod(D_tri)
T_{JI_set} ← MultiOtsuMethod(JI)
/* Compute cross product of the sets for possible
   combinations                                        */
T_comb ← CrossProd(T_{both_set}, T_{any_set}, T_{tri_set}, T_{JI_set})
/* Calculate the optimum threshold                     */
RL_Ratio ← Φ
forall (T_both, T_JI, T_any, T_tri) ∈ T_comb do
    right ← max(min(getRight(T_both/2), getRight(T_JI/2)),
    getRight(T_any), getRight(T_tri/2))
    left ← |E| − right
    RL_Ratio ← RL_Ratio ∪ (1 − right/left)
/* Choose combination for which value of RL_Ratio is
   minimum                                             */
(T*_both, T*_JI, T*_any, T*_tri) ←
 getOptimumThreshold(T_comb, RL_Ratio)
/* Classify the edges.                                 */
B100 ← Φ; B0 ← Φ;
forall e ∈ E do
    if ((D_both(e) > T*_both/2 AND JI(e) > T*_JI/2) OR
    (D_any(e) > T*_any OR D_tri(e) > T*_tri/2)) then
        B100 ← B100 ∪ e
    else
        B0 ← B0 ∪ e
```

Classification using multi-Otsu (Algorithm 2). Applying Multi-Otsu to test all combinations of the thresholds and features is computationally expensive. For k thresholds and 4 features, $\mathcal{O}(k^4)$ combinations should be checked. For computationally feasible classification, we compute the thresholds for each feature separately, rather than in combination.

For each threshold, we group edges based on whether they are higher or lower than the threshold. Given a strong community structure, the number of edges within communities should be at least as much as the number of edges across communities. As a first cut, we identify the optimal threshold as one that produces an almost equal number of edges on the left (lower) and right (higher) sides.

Classification using iterative multi-Otsu. The thresholds obtained by Algorithm 2 provide a lower bound on the number of within-community edges. To further improve the accuracy, we apply Multi-Otsu iteratively on the edges in **B0**. We apply Algorithm 2 to find the optimal threshold on the edges remaining in **B0** and obtain a new division of **B0** and **B100**. We continue this iteration until the change in threshold is $\leq \delta$, (we set $\delta = 0.01$) and no new edges move from **B0** to **B100**. We call this method *Multi-Otsu iterative*.

Handling singleton communities The subgraph indu-ced by the edges in **B100** provides the constant communities. Some communities may be singletons, i.e. composed of one vertex. For a singleton vertex of degree 2, if both the neighbors are in the same community, we move the vertex to that community. If the neighbors are in different communities, the vertex is moved to any one of the communities.

This heuristic gives a slightly higher F1-score, but the accuracy decreases if applied to vertices of higher degree. This is because the probability of a singleton being in a community decreases with the number of neighboring communities.

IV. Experimental Setup

Datasets and Ground Truth. We used a set of real-world networks as given in Table I. To obtain ground truth, we executed 50 runs of a community detection method per network. At each execution, we permuted vertex order to change the community results. The communities that were common to all of these runs were designated as the constant community ground truth, for the given community detection method.

For small networks, we created ground truth for three community detection algorithms; the Louvain method [4], the Infomap method [5] and the Label Propagation method [6].

Executing multiple runs was too expensive on medium (**10K+ nodes** and **10K+** to **1M+ edges**) and large (**1M+ nodes** and **1M+** to **10M+ edges**) networks. We created ground truths only for Louvain, the fastest of the three methods.

Name	Vertices	Edges
	Small-size n/ws	
Football [7]	115	1226
Jazz [7]	198	2742
Dolphin [7]	62	159
Email [8]	1133	5451
Karate club [7]	34	77
Polbooks [7]	105	441
	Medium-size n/ws	
Co-authorship [9]	103,677	352,183
Com-dblp [10]	317,080	1,049,866
Com-amazon [10]	334,863	925,872
	Large-size n/ws	
Com-Youtube [10]	1,134,890	2,987,624
Com-LiveJournal [10]	3,997,962	34,681,189
Wiki-topcats [10]	1,791,489	28,511,807

TABLE I: The test suite of real-world networks.

V. Empirical Results

We compare our proposed methods: *(i)* Binary-Otsu; *(ii)* multi-level Otsu thresholding (Multi-Otsu); *(iii)* iterative multi-

Method	GT	Small-size graphs (F1-Scores)					
		Football	Jazz	Email	Dolphin	Karate	Polbooks
Binary-Otsu	Louvain	0.97	0.78	0.73	0.88	0.82	0.87
	Infomap	0.97	0.92	0.73	0.89	0.90	0.88
	LP	0.96	0.89	0.72	0.79	0.90	0.89
Multi-Otsu	Louvain	0.96	0.81	0.73	0.88	0.84	0.88
	Infomap	0.93	0.81	0.70	0.86	0.93	0.88
	LP	0.92	0.84	0.68	0.87	0.94	0.89
Multi-Otsu iterative	Louvain	0.96	0.82	0.75	0.88	0.85	0.88
	Infomap	0.94	0.81	0.70	0.87	0.94	0.88
	LP	0.93	0.84	0.68	0.87	0.94	0.89
Multi-Otsu iterative+SC	Louvain	0.96	0.82	0.75	0.88	0.85	0.88
	Infomap	0.94	0.79	0.70	0.87	0.94	0.88
	LP	0.93	0.84	0.68	0.87	0.94	0.89
Consensus (cp=2%)	Louvain	0.92	0.74	0.30	0.85	0.78	0.71
	Infomap	0.99	0.79	0.77	0.87	0.94	0.89
	LP	0.91	0.88	0.37	0.70	0.52	0.76
Consensus (cp=4%)	Louvain	0.90	0.74	0.29	0.85	0.78	0.71
	Infomap	0.99	0.80	0.77	0.87	0.94	0.89
	LP	0.92	0.88	0.36	0.71	0.52	0.76
Consensus (cp=8%)	Louvain	0.90	0.74	0.29	0.85	0.77	0.71
	Infomap	0.97	0.79	0.77	0.87	0.94	0.89
	LP	0.92	0.87	0.36	0.70	0.53	0.76
CHAMP	Louvain	0.97	0.88	0.74	0.89	0.87	0.87
	Infomap	0.15	0.54	0.64	0.34	0.40	0.65
	LP	0.61	0.65	0.23	0.38	0.6	0.47

TABLE II: Performance of different methods for obtaining constant communities for small networks. Green cells indicate the highest F1-scores results, blue cells indicate the second highest and red cells indicate the worst results (for Consensus method cp is the *convergence percentage*; in Multi-Otsu iterative+SC, SC stands for *singleton communities*).

Method	GT	Medium- and large-size graphs (F1-scores)					
		com-dblp	coauthorship	com-amazon	com-youtube	wiki-topcats	com-liveJ
Binary-Otsu	Louvain	0.75	0.89	0.73	0.54	0.39	0.54
Multi-Otsu	Louvain	0.75	0.91	0.87	0.55	0.61	0.75
Multi-Otsu iterative	Louvain	0.75	0.91	0.87	0.68	0.63	0.78
Multi-Otsu iterative+SC	Louvain	0.75	0.91	0.87	0.81	0.65	0.79
Consensus(2%)	Louvain	X	0.70	X	X	X	X
CHAMP	Louvain	X	X	X	X	X	X

TABLE III: Performance of different methods for obtaining constant communities for million scale networks. Green cells represent the highest F1-scores, blue cells indicate the second highest and red cells indicate worst results. X: The process did not end within a sizable amount of time. In Multi-Otsu iterative+SC, SC stands for *singleton communities*.

level Otsu thresholding (Multi-Otsu iterative) with these **baseline methods**; the consensus community algorithm in [11] and the CHAMP method from [12]. The average results from several runs are reported. The algorithms were executed on an IntelXeon(R) Processor with CPU E3-1270 V2, speed 3.50GHz×8, and 32 GB Memory. All relevant codes are at https://github.com/anjangit000/ImgThAlgoConsComm

Accuracy with respect to the ground truth. We compared the set of edges classified as part of constant communities with the edges that are internal to constant communities as per the ground truth. Table II shows the F1-score for the small networks and Table III for medium and large networks. The results show that **our methods almost always produce the highest F1-score** for all three algorithms.

Overlap with the ground truth communities. We obtained the connected components induced by the edges predicted to be in constant communities. These components form the predicted constant communities. We compute the NMI between these predicted constants and those obtained from the ground truth. Table IV shows that **our results are comparable or better** than the baseline methods.

Execution time. Table V shows the execution time for obtaining constant communities using our algorithms and the baselines for the small networks. Our algorithms are $\approx$ **10 times faster**. For medium and large networks (see Table V), our algorithms are fast even for networks with 10M+ nodes and 100M+ edges. The baseline methods take either a huge time to finish or do not complete within a feasible time frame.

VI. Related work

Community detection is well-studied and numerous algorithms exist (see survey [13]). However, finding constant communities is less studied. Newman [14] shows that clusterings leading to optimum parameter values are similar and built from nodes that are usually found together in the same community. In [1] properties of constant communities are studied with respect to within community and across community edges. In [15] authors use a greedy approach(consensus matrix) to produce consistent and stable partitions. Variations include [11], [12], [16]–[20]. Nevertheless, as seen here, these are not yet fast for large networks.

VII. Conclusions and future work

We developed an efficient algorithm, using variants of the image thresholding techniques to classify the edges, for identifying constant communities that scale to large networks.

Method	Networks										
	football	**jazz**	**email**	**dolphin**	**polbook**	**karate**	**com-amazon**	**com-dblp**	**coauthorship**	**youtube**	**wiki-top**
Our method with best F1	0.97	0.82	0.71	0.86	0.85	0.91	0.82	0.79	0.80	0.76	0.67
Consensus with best F1	0.96	0.77	0.77	0.86	0.80	0.90	X	X	0.63	X	X
CHAMP with best F1	0.97	0.83	0.71	0.77	0.79	0.91	X	X	X	X	X

TABLE IV: Comparison of NMI. Best results are highlighted in green and the worst in red.

Method	All networks					
	com-dblp	**co-author-ship**	**com-ama-zon**	**com-youtube**	**wiki-topcats**	**com-liveJ**
Binary-Otsu	19.26s	6.17s	1m10s	8m12s	16h04m	10h23m
Multi-Otsu	23.29s	6.47s	1m12s	10m12s	17h03m	12h21m
Multi-Otsu iterative	24.25s	7.13s	1m14s	23m06s	27h03m	24h21m
Multi-Otsu iterative+SC	24.25s	7.13s	1m19s	26m41s	28h10m	25h21m
BF	2h26m	5h25m	34h8m	140h8m	872h10m	963h3m
Consensus (cp=2%)	X	141m	X	X	X	X
CHAMP	X	X	X	X	X	X

TABLE V: Time for identifying constant communities for some networks. Best timing performances for each dataset are denoted by green cells and the worst by red cells. X: The process did not end within a sizable amount of time. In Multi-Otsu iterative+SC, SC stands for *singleton communities*. BF stands for bruteforce method also used to generate our Groud truth(#iterations = 50).

We plan to apply the constant communities to various downstream applications, including outlier detection, domain adaptation, and feature selection. We also aim to parallelize our algorithms to further improve the performance.

Acknowledgement. S.B.'s work was supported by NSF CCF# 1956373.

REFERENCES

[1] T. Chakraborty, S. Srinivasan, N. Ganguly, S. Bhowmick, and A. Mukherjee, "Constant communities in complex networks," *Scientific Reports*, vol. 3, no. 1, 2013.
[2] N. Otsu, *IEEE Transactions on Systems, Man and Cybernetics*, no. 1.
[3] D.-Y. Huang and C.-H. Wang, "Optimal multi-level thresholding using a two-stage otsu optimization approach," *Pattern Recognition Letters*, vol. 30, pp. 275–284, February 2009.
[4] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. of Stat. Mech.*, no. 10, p. P10008, 2008.
[5] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *PNAS*, vol. 105, no. 4, pp. 1118–1123, 2008.
[6] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Phys. Rev. E*, vol. 76, no. 3, Sep 2007.
[7] S. P. Kolodziej, M. Aznaveh, M. Bullock, J. David, T. A. Davis, M. Henderson, Y. Hu, and R. Sandstrom, "The suitesparse matrix collection website interface," *Journal of Open Source Software*, vol. 4, no. 35, p. 1244, 2019.
[8] R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt, and A. Arenas, "Self-similar community structure in a network of human interactions," *Phys. Rev. E*, vol. 68, no. 6, Dec 2003.
[9] T. Chakraborty, S. Srinivasan, N. Ganguly, A. Mukherjee, and S. Bhowmick, "On the permanence of vertices in network communities." in *KDD*. ACM, 2014, pp. 1396–1405.
[10] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, Jun. 2014.
[11] A. Tandon, A. Albeshri, V. Thayananthan, W. Alhalabi, and S. Fortunato, "Fast consensus clustering in complex networks," *Phys. Rev. E*, no. 4, Apr 2019.
[12] W. Weir, S. Emmons, R. Gibson, D. Taylor, and P. Mucha, "Post-processing partitions to identify domains of modularity optimization," *Algorithms*, vol. 10, no. 3.
[13] "A comprehensive literature review on community detection: Approaches and applications," *Procedia Computer Science*.
[14] M. A. Riolo and M. E. J. Newman, "Consistency of community structure in complex networks," *Phys. Rev. E*, vol. 101, no. 5, May 2020.
[15] A. Lancichinetti and S. Fortunato, "Consensus clustering in complex networks," *Sci. Rep.*, vol. 2, no. 1, Mar 2012.
[16] L. Jeub, O. Sporns, and S. Fortunato, "Multiresolution consensus clustering in networks," *Scientific Reports*, vol. 8, February 2018.
[17] V. Poulin and F. Théberge, "Ensemble clustering for graphs: comparisons and applications," *Applied Network Science*, vol. 4, no. 1, p. 51, 2019.
[18] T. Chakraborty, N. Park, and V. S. Subrahmanian, "Ensemble-based algorithms to detect disjoint and overlapping communities in networks," pp. 73–80, Aug 2016.
[19] T. Chakraborty, "On the discovery of invariant groups in complex networks," *Journal of Complex Networks*, vol. 5, pp. 734–749, 10 2017.
[20] N. Nguyen, A. Alim, T. Dinh, and M. Thai, "A method to detect communities with stability in social network," *Social Network Analysis and Mining*, vol. 4, 08 2014.