# A New Measure of Modularity in Hypergraphs: Theoretical Insights and Implications for Effective Clustering

Tarun Kumar[1,2](✉), Sankaran Vaidyanathan[3], Harini Ananthapadmanabhan[2], Srinivasan Parthasarathy[4], and Balaraman Ravindran[1,2]

[1] Robert Bosch Centre for Data Science and AI (RBCDSAI), Chennai, India
[2] Department of Computer Science and Engineering, IIT Madras, Chennai, India
tkumar@cse.iitm.ac.in
[3] College of Information and Computer Sciences, University of Massachusetts Amherst, Amherst, USA
[4] Department of Computer Science and Engineering, The Ohio State University, Columbus, USA

**Abstract.** Many real-world systems consist of entities that exhibit complex group interactions rather than simple pairwise relationships; such multi-way relations are more suitably modeled using hypergraphs. In this work, we generalize the framework of modularity maximization, commonly used for community detection on graphs, for the hypergraph clustering problem. We introduce a hypergraph null model that can be shown to correspond exactly to the configuration model for undirected graphs. We then derive an adjacency matrix reduction that preserves the hypergraph node degree sequence, for use with this null model. The resultant modularity function can be maximized using the Louvain method, a popular fast algorithm known to work well in practice for graphs. We additionally propose an iterative refinement over this clustering that exploits higher-order information within the hypergraph, seeking to encourage balanced hyperedge cuts. We demonstrate the efficacy of our methods on several real-world datasets.

## 1 Introduction

While most approaches for learning clusters on graphs assume pairwise (or dyadic) relationships between entities, many entities in real world network systems engage in more complex, multi-way (super-dyadic) relations. *Hypergraphs* provide a natural representation for such super-dyadic relations; for example, in a co-citation network, a hyperedge could represent a group of co-cited papers. Indeed, learning on hypergraphs has been gaining recent traction [7,21,25,26]. Analogous to the graph clustering task, *Hypergraph clustering* seeks to find dense

T. Kumar and S. Vaidyanathan—Equal contribution.
S. Vaidyanathan—Work done while the author at IIT Madras.
H. Ananthapadmanabhan—Currently at Google, Bangalore, India.

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2019, SCI 881, pp. 286–297, 2020.
https://doi.org/10.1007/978-3-030-36687-2_24

connected components within a hypergraph [19]. This has been applied to varied problems such as VLSI placement [11], image segmentation [13], and modeling eco-biological systems [6], among others.

A few previous works on hypergraph clustering [1,14,15,20,23] have limited their focus to k-uniform hypergraphs, where all hyperedges have the same fixed size. [27] extends the Spectral Clustering framework for general hypergraphs by proposing a suitable hypergraph Laplacian, which implicitly defines a reduction of the hypergraph to a graph [2,16].

*Modularity maximization* [17] is an alternative methodology for clustering on graphs, which additionally provides a useful metric for measuring cluster quality in the *modularity* function and return the number of clusters automatically. In practice, a greedy fast and scalable optimization algorithm known as the Louvain method [3] is commonly used.

However, extending the modularity function to hypergraphs is not straightforward. One approach would be to reduce a hypergraph to a simple graph using a clique expansion and then employ a standard modularity-based solution. Such an approach would lose critical information encoded within the super-dyadic hyperedge structure. A clique expansion would also not preserve the hypergraph's node degrees, which are required for the null model that modularity maximization methods are based on. Encoding the hyperedge-centric information present within the hypergraph is key to the development of an appropriate modularity-based framework for clustering.

Additionally, when viewing the clustering problem via a minimization function (analogous to minimizing the cut), there are multiple ways to cut a hyperedge. Based on the proportion and assignments of nodes on different sides of the cut, the clustering will change. One way of incorporating information based on properties of hyperedges or their vertices is to introduce hyperedge weights based on a metric or function of the data. Building on this idea, we make the following contributions in this work:

- We define a null model on hypergraphs, and prove its equivalence to the configuration model [18] for undirected graphs. We derive a node-degree preserving graph reduction to satisfy this null model. Subsequently, we define a modularity function using the above which can be maximized using the Louvain method.
- We propose an iterative hyperedge reweighting procedure that leverages information from the hypergraph structure and the balance of hyperedge cuts.
- We empirically evaluate the resultant algorithm, titled *Iteratively Reweighted Modularity Maximization* (IRMM), on several real-world datasets and demonstrate its efficacy and efficiency over competitive baselines.

## 2 Background

### 2.1 Hypergraphs

Let $G = (V, E, w)$ be a hypergraph, with vertex set $V$ and hyperedge set $E$. Each hyperedge can be associated with a positive weight $w(e)$. Degree of a vertex $v$

is denoted as $d(v) = \sum_{e \in E, v \in e} w(e)$. The degree of a hyperedge $e$ is the number of nodes it contains; denoted by $\delta(e) = |e|$. We denote the number of vertices as $n = |V|$ and the number of edges as $m = |E|$. The incidence matrix $H$ is given by $h(v, e) = 1$ if vertex $v$ is in hyperedge $e$, and 0 otherwise. $W$ is the hyperedge weight matrix and $D_e$ is the edge degree matrix; these are diagonal matrices of size $m \times m$. $D_v$ is the vertex degree matrix, of size $n \times n$.

**Clique Reduction:** For any hypergraph, one can compute its *clique reduction* [9] by replacing each hyperedge by a clique formed from its node set. The adjacency matrix for the clique reduction of a hypergraph with incidence matrix $H$ is $A = HWH^T$. $D_v$ may be subtracted from this matrix to remove self-loops.

### 2.2 Modularity

Modularity [17] is a metric of clustering quality that measures whether the number of within-cluster edges is greater than its expected value. This is defined as:

$$Q = \frac{1}{2m} \sum_{ij} [A_{ij} - P_{ij}] \delta(g_i, g_j) \tag{1}$$

where $P_{ij}$ is denotes the expected number of edges between nodes $i$ and $j$. The *configuration model* [18] used for graphs produces random graphs with a fixed degree sequence, by drawing random edges such that the node degrees are preserved. For two nodes $i$ and $j$, with degrees $k_i$ and $k_j$ respectively, we have:

$$P_{ij} = \frac{k_i k_j}{\sum_{j \in V} k_j}$$

## 3 Hypergraph Modularity

We propose a simple but novel *node-degree preserving* null model for hypergraphs. Analogous to the configuration model for graphs, the sampling probability for a node is proportional to the number (or in the weighted case, the total weight) of hyperedges it participates in. Specifically, we have:

$$P_{ij}^{hyp} = \frac{d(i) \times d(j)}{\sum_{v \in V} d(v)} \tag{2}$$

The above null model preserves the node degree sequence of the original hypergraph. When using this null model to define modularity, we get the expected number of hyperedges that two nodes $i$ and $j$ participate in. However, while taking the clique reduction, the degree of a node in the corresponding graph is not the same as its degree in the original hypergraph, as verified below.

**Lemma 1.** *For the clique reduction of a hypergraph with incidence matrix $H$, the degree of a node $i$ in the reduced graph is given by:*

$$k_i = \sum_{e \in E} H(i, e) w(e) (\delta(e) - 1)$$

*Proof.* The adjacency matrix of the reduced graph is given by $A_{clique} = HWH^T$,

$$(HWH^T)_{ij} = \sum_{e \in E} H(i,e)w(e)H(j,e)$$

Note that we do not have to consider self-loops, since they are not cut during the modularity maximization process. This is done by explicitly setting $A_{ii} = 0$ for all $i$. We can write the degree of a node $i$ in the reduced graph as:

$$\begin{aligned} k_i = \sum_j A_{ij} &= \sum_j \sum_{e \in E} H(i,e)w(e)H(j,e) \\ &= \sum_{e \in E} H(i,e)w(e) \sum_{j:j \neq i} H(j,e) \\ &= \sum_{e \in E} H(i,e)w(e)(\delta(e) - 1) \end{aligned}$$

As shown above, the node degree is over counted by a factor of $(\delta(e) - 1)$ for each hyperedge $e$. We can hence correct it by scaling down each $w(e)$ by a factor of $(\delta(e) - 1)$. This leads to the following corrected adjacency matrix:

$$A^{hyp} = HW(D_e - I)^{-1}H^T \tag{3}$$

**Proposition 1.** *For the reduction of a hypergraph given by the adjacency matrix $A = HW(D_e - I)^{-1}H^T$, the degree of a node $i$ in the reduced graph (denoted $k_i$) is equal to its hypergraph node degree $d(i)$.*

*Proof.* We have,

$$(HW(D_e - I)^{-1}H^T)_{ij} = \sum_{e \in E} \frac{H(i,e)w(e)H(j,e)}{\delta(e) - 1}$$

Note again that we do not have to consider self-loops, since they are not cut during the modularity maximization process (explicitly setting $A_{ii} = 0$ for all $i$). We can rewrite the degree of a node $i$ in the reduced graph as

$$\begin{aligned} k_i = \sum_j A_{ij} &= \sum_{e \in E} \frac{H(i,e)w(e)}{\delta(e) - 1} \sum_{j:j \neq i} H(j,e) \\ &= \sum_{e \in E} H(i,e)w(e) = d(i) \end{aligned}$$

We can use this node-degree preserving reduction, with the diagonals zeroed out, to implement the null model from Eq. 2. As in Eq. 1, we can obtain an expression for the hypergraph modularity, which can be maximized using a Louvain-style algorithm.

$$Q^{hyp} = \frac{1}{2m} \sum_{ij} [A^{hyp}_{ij} - P^{hyp}_{ij}] \delta(g_i, g_j) \tag{4}$$

As with any weighted graph, the range of this function is $[-1, 1]$. We would get $Q^{hyp} = -1$ when no pair of nodes in a hyperedge belong to the same cluster, and $Q^{hyp} = 1$ when any two nodes that are part of the same hyperedge are always part of the same cluster. $Q^{hyp} = 0$ when, for any pair of nodes $i$ and $j$, the number of hyperedges that contain both $i$ and $j$ is equal to the number of randomly wired hyperedges containing both $i$ and $j$, given by the null model.

## 4 Iterative Hyperedge Reweighting

When improving clustering, we look at minimizing the number of between-cluster edges that get cut, which for a hypergraph is given by the total volume of the hyperedge cut. We first consider the two-cluster case as in [27], where the set $V$ is partitioned into two clusters, $S$ and $S^c$. For a given hyperedge $e$, the volume of the cut is proportional to $|e \cap S||e \cap S^c|$, which gives the number of cut sub-edges when the hyperedge is reduced to a clique. This is minimized when all vertices of $e$ go into one partition. A min-cut algorithm would hence favour cuts that are as unbalanced as possible [10].

For a given hyperedge, if there were a larger portion of its vertices in one cluster and a smaller portion in the other, it is likely that the smaller group of vertices are actually similar to the rest and should be pulled into the larger cluster. Similarly, the vertices in a hyperedge that is cut equally between clusters are equally likely to lie in either of the clusters. We would hence want unbalanced hyperedges to be retained in clusters and the more balanced hyperedges to be cut. This can be done by increasing the weights of hyperedges that get unbalanced cuts, and decreasing the weights of hyperedges that get more balanced cuts.

Considering the case where a hyperedge is partitioned into two, for a cut hyperedge with $k_1$ and $k_2$ nodes in each partition ($k_1, k_2 \neq 0$), we have the following equation that operationalizes the aforementioned scheme:

$$t = \left(\frac{1}{k_1} + \frac{1}{k_2}\right) \times \delta(e) \tag{5}$$

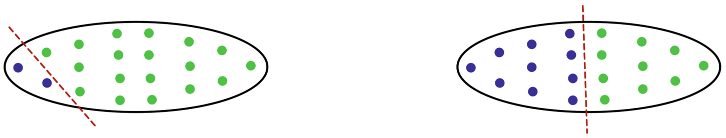
For two partitions, $\delta(e) = k_1 + k_2$. In Fig. 4, note that $t$ is minimized when $k_1 = k_2 = \delta(e)/2$, which gives $t = 4$. We can then generalize Eq. 5 to $c$ partitions as follows:

$$w'(e) = \frac{1}{m} \sum_{i=1}^{c} \frac{1}{k_i + 1} [\delta(e) + c] \tag{6}$$

Here, both the $+1$ and $+c$ terms are added for smoothing, to account for cases where any of the $k_i$'s are zero. We divide by $m$ to normalize the weights (Fig. 1).

Let $w_t(e)$ be the weight of hyperedge $e$ in the $t^{th}$ iteration, and $w'(e)$ be the weight computed at a given iteration then weight update rule can be written as:

$$w_{t+1}(e) = \alpha w_t(e) + (1 - \alpha) w'(e) \tag{7}$$



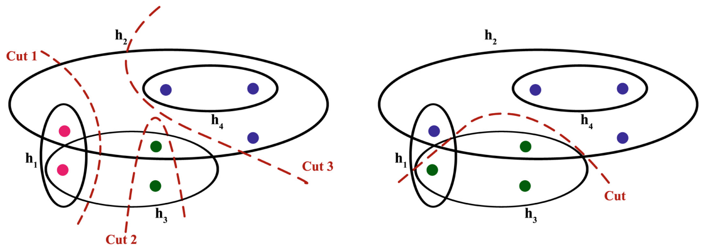
$$t = \left(\frac{1}{2} + \frac{1}{18}\right) \times 20 = 11.111 \qquad t = \left(\frac{1}{10} + \frac{1}{10}\right) \times 20 = 4$$

**Fig. 1.** Reweighting for different hyperedge cuts

## 4.1 A Simple Example

The following example illustrates the effect of a single iteration of hyperedge reweighting. As seen in Fig. 2, the initial clustering of this hypergraph resulted in two highly unbalanced cuts. Cut 1 and Cut 2 each split hyperedge $h_2$ in a 1:4 ratio, and also each split hyperedge $h_3$ in a $1:2$ ratio respectively. Cut 3 splits hyperedge $h_1$ in a 2:3 ratio.

After reweighting, the 1:4 splits are removed. This reduces the number of cuts from 3 to just 1, leaving two neat clusters. The single nodes in $h_1$ and $h_3$, initially assigned to another cluster, have been pulled back into their respective (larger) clusters. This captures the intended behaviour for the re-weighting scheme as described earlier.



(a) Before Reweighting (b) After Reweighting

**Fig. 2.** Effect of iterative reweighting

## 5 Evaluation on Ground Truth

We use the average F1 measure [24] and Rand Index scores to evaluate clustering performance on real-world data with ground truth class labels. Average F1 scores are obtained by computing the F1 score of the best matching ground-truth class to the observed cluster, and the F1 score of the best matching observed cluster to the ground truth class, then averaging the two scores. The proposed methods are shown in the results table as *Hypergraph-Louvain* and *IRMM*.

**Number of Clusters Returned:** While implementing the modularity function defined in Eq. 4, we use the Louvain method to find clusters by maximizing the hypergraph modularity. By default, the algorithm returns the number of clusters. To return a fixed number of clusters $c$, we used hierarchical clustering with the average linkage criterion as a post-processing step.

**Settings for IRMM:** We tuned hyperparameter $\alpha$ over the set of values $0.1, 0.2, ..., 0.9$. We did not find considerable difference in the resultant F1 scores, and minimal difference in the rate of convergence, over a wide range of values. As $\alpha$ is a scalar coefficient in a moving average, it did not result in a large difference in resultant weight values when set in this useful range. Hence, for the experiments, we chose to leave it at $\alpha = 0.5$. For the iterations, we set the stopping threshold for the weights at 0.01.

## 5.1 Compared Methods

**Clique Reductions:** We took the clique reduction of the hypergraph and ran the graph versions of Spectral Clustering and the Louvain method. These are referred to as *Clique-Spectral* and *Clique-Louvain* respectively.

**Hypergraph Spectral Clustering:** Here, the top $c$ eigenvectors of the Hypergraph Laplacian (as defined in [27]) were found, and then clustered using bisecting k-means. We refer to this method as *Hypergraph-Spectral*.

**hMETIS** [12] **and PaToH** [5]**:** These are hypergraph partitioning algorithms that are commonly used. We used the codes provided by the respective authors.

## 5.2 Datasets

For all datasets, we took the single largest connected component of the hypergraph. In each case, the class labels were taken as ground truth clusters. More statistics on the datasets are given in Table 1.

**Table 1.** Dataset description

| Dataset | # nodes | # hyperedges | Avg. hyperedge degree | Avg. node degree | # classes |
|---|---|---|---|---|---|
| TwitterFootball | 234 | 3587 | 15.491 | 237.474 | 20 |
| Cora | 2708 | 2222 | 3.443 | 2.825 | 7 |
| Citeseer | 3264 | 3702 | 27.988 | 31.745 | 6 |
| MovieLens | 3893 | 4677 | 79.875 | 95.961 | 2 |
| Arnetminer | 21375 | 38446 | 4.686 | 8.429 | 10 |

**MovieLens** [4]**:** We used the *director* relation to define hyperedges and build a co-director hypergraph, where the nodes represent movies. A group of nodes are connected by a hyperedge if they were directed by the same individual.

**Cora and Citeseer**: In these datasets, nodes represent papers. The nodes are connected by a hyperedge if they share the same set of words [22].

**TwitterFootball:** This is taken from one view of the Twitter dataset [8]. It represents members of 20 different football clubs of the English Premier League. Here, the nodes are the players, and hyperedges are formed based on whether they are co-listed.

**Arnetminer:** In this significantly larger co-citation network, the nodes represent papers and hyperedges are formed between co-cited papers. We used the nodes from the CS discipline, and its 10 sub-disciplines were treated as clusters.

## 5.3 Experiments

We compare the average F1 and Rand Index (RI) scores for the different datasets on all the given methods. For Louvain methods, the number of clusters was returned by the algorithm in an unsupervised manner and the same number was used for spectral methods. The results are given in Table 2. We also ran the same experiments with the number of clusters set to the number of ground truth classes, using the postprocessing methodology described earlier (Table 3).

## 5.4 Results

In both experiment settings, *IRMM* shows the best average F1 scores on all datasets, and the best Rand Index scores on all but one dataset. Additionally, both hypergraph modularity maximization methods show competitive performance with respect to the baselines. Figure 3 shows the results for varying number of clusters. *Clique-Louvain* sometimes returned a lower number of clusters than IRMM, and hence these curves are shorter in some of the plots. On TwitterFootball, *IRMM* returns fewer than ground truth clusters; and hence the corresponding entry in Table 3 is left blank.

On some datasets, the best performance is achieved when the number of clusters returned by the Louvain method is used (e.g Citeseer, Cora), and on others when the ground truth number of classes is used (e.g ArnetMiner, MovieLens). This could be based on the structure of clusters in the network and its relationship to the ground truth classes. A class could have comprised multiple smaller clusters, which were detected by the Louvain algorithm. In other cases, the cluster structure could have corresponded better to the class labels.

It is evident that Hypergraph based methods outperform the respective clique reduction methods on all datasets and both experiment settings. We infer that super-dyadic relational information captured by the hypergraph has a positive impact on the clustering performance.

**Table 2.** Average F1 and RandIndex scores; no. of clusters returned by Louvain method

| | Citeseer | | Cora | | MovieLens | | TwitterFootball | | Arnetminer | |
|---|---|---|---|---|---|---|---|---|---|---|
| | F1 | RI | F1 | RI | F1 | RI | F1 | RI | F1 | RI |
| hMETIS | 0.1087 | 0.6504 | 0.1075 | 0.7592 | 0.1291 | 0.4970 | 0.3197 | 0.7639 | 0.0871 | 0.0416 |
| PaToH | 0.0532 | 0.6612 | 0.1171 | 0.6919 | 0.1104 | 0.4987 | 0.1132 | 0.7553 | 0.0729 | 0.0052 |
| Clique Spectral | 0.1852 | 0.7164 | 0.1291 | 0.2478 | 0.1097 | 0.4806 | 0.4496 | 0.7486 | 0.0629 | 0.0610 |
| Hypergraph Spectral | 0.2774 | **0.8210** | 0.2517 | 0.5743 | 0.118 | 0.4977 | 0.5055 | 0.9016 | 0.0938 | 0.0628 |
| Clique Louvain | 0.1479 | 0.7361 | 0.2725 | 0.7096 | 0.1392 | 0.4898 | 0.2238 | 0.6337 | 0.1378 | 0.0384 |
| Hypergraph Louvain | 0.2782 | 0.7899 | 0.3248 | 0.8238 | 0.1447 | 0.4988 | 0.5461 | 0.9056 | 0.1730 | 0.0821 |
| IRMM | **0.4019** | 0.7986 | **0.3709** | **0.8646** | **0.1963** | **0.5091** | **0.5924** | **0.9448** | **0.1768** | **0.0967** |

**Table 3.** Avg. F1 and RandIndex scores; no. clusters set to no. of ground truth classes

| | Citeseer | | Cora | | MovieLens | | TwitterFootball | | Arnetminer | |
|---|---|---|---|---|---|---|---|---|---|---|
| | F1 | RI | F1 | RI | F1 | RI | F1 | RI | F1 | RI |
| hMETIS | 0.1451 | 0.6891 | 0.2611 | 0.7853 | 0.4445 | 0.5028 | 0.3702 | 0.7697 | 0.3267 | 0.3116 |
| PaToH | 0.0710 | 0.7312 | 0.1799 | 0.7208 | 0.3239 | 0.4984 | 0.1036 | 0.7618 | 0.2756 | 0.182 |
| Clique Spectral | 0.2917 | 0.7369 | 0.2305 | 0.3117 | 0.2824 | 0.4812 | 0.4345 | 0.7765 | 0.387 | 0.3762 |
| Hypergraph Spectral | 0.3614 | **0.8267** | 0.2672 | 0.5845 | 0.3057 | 0.5006 | 0.5377 | 0.9112 | 0.4263 | 0.3851 |
| Clique Louvain | 0.1479 | 0.7361 | 0.2725 | 0.7096 | 0.2874 | 0.4982 | 0.2238 | 0.6337 | 0.4587 | 0.4198 |
| Hypergraph Louvain | 0.3491 | 0.8197 | 0.3314 | 0.8441 | 0.3411 | 0.5119 | 0.5461 | 0.9056 | 0.4948 | 0.5359 |
| IRMM | **0.4410** | 0.8245 | **0.3966** | **0.889** | **0.4445** | **0.5347** | – | – | **0.5299** | **0.5506** |

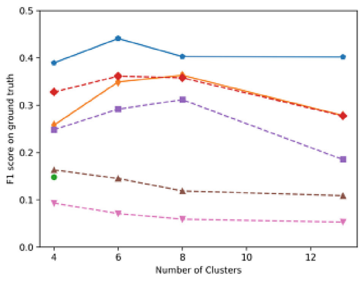
## 5.5 Effect of Reweighting on Hyperedge Cuts

The plots in Fig. 4 illustrate the effect of hyperedge reweighting over iterations. We found the relative size of the largest partition of each hyperedge, and binned them in intervals of relative size = 0.1. The plot shows the fraction of hyperedges that fall in each bin over each iteration.

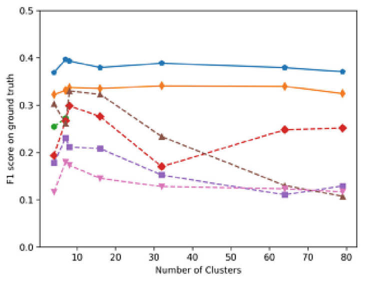
$$\text{relative size}(e) = \max_i \frac{\text{number of nodes in cluster } i}{\text{number of nodes in the hyperedge } e}$$

We refer these hyperedges as *fragmented* if the relative size of its largest partition is lower than a threshold (here set at 0.3), and *dominated* of the relative size of its largest partition is higher than the threshold. The fragmented edges are likely to be balanced, since the largest cluster size is low.

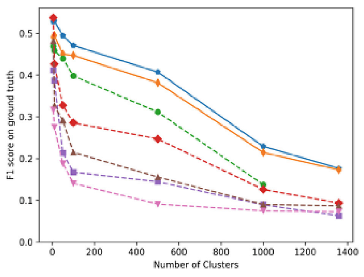
On the smaller TwitterFootball dataset, which has a greater number of ground truth classes, we see that the number of dominated edges decreases and the number of fragmented edges increases. This is as expected; the increase in fragmented edges is likely to correspond to more balanced cuts. A similar trend is reflected in the larger Cora dataset.

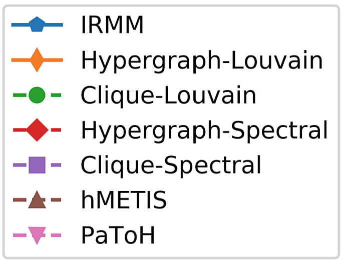


(a) Citeseer



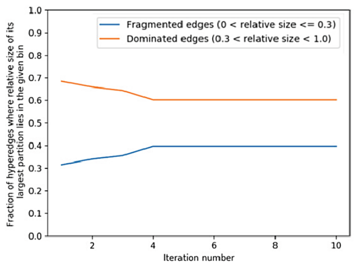
(b) Cora



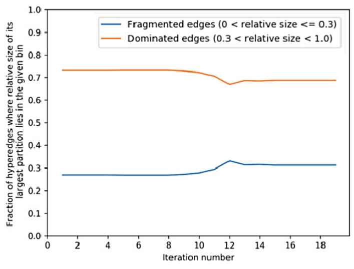


(c) Arnetminer

**Fig. 3.** Symmetric F1 scores for varying number of clusters



(a) TwitterFootball



(b) Cora

**Fig. 4.** Reweighting for different hyperedge cuts.

## 6 Conclusion

In this work, we have considered the problem of modularity maximization on hypergraphs. In presenting a modularity function for hypergraphs, we derived a node degree preserving graph reduction and a hypergraph null model. To refine the clustering further, we proposed a hyperedge reweighting procedure that balances the cuts induced by the clustering method. Empirical evaluations on real-world data illustrated the performance of our resultant method, entitled

Iteratively Reweighted Modularity Maximization (IRMM). We leave the exploration of additional constraints and hyperedge-centric information in the clustering framework for future work.

**Acknowledgements.** This work was partially supported by Intel research grant RB/18-19/CSE/002/INTI/BRAV to BR.

## References

1. Agarwal, S., Lim, J., Zelnik-Manor, L., Perona, P., Kriegman, D., Belongie, S.: Beyond pairwise clustering. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), vol. 2, pp. 838–845, June 2005
2. Agarwal, S., Branson, K., Belongie, S.: Higher order learning with graphs. In: ICML 2006: Proceedings of the 23rd International Conference on Machine Learning, pp. 17–24 (2006)
3. Blondel, V.D., loup G., J., L., R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech.: Theory Exp. (10), P10008 (2008)
4. Cantador, I., Brusilovsky, P., Kuflik, T.: 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In: Proceedings of the 5th ACM Conference on Recommender Systems, RecSys 2011. ACM, New York (2011)
5. Çatalyürek, Ü., Aykanat, C.: PaToH (partitioning tool for hypergraphs), pp. 1479–1487. Springer, Boston (2011). https://doi.org/10.1007/978-0-387-09766-4_93
6. Estrada, E., Rodriguez-Velazquez, J.A.: Complex networks as hypergraphs. arXiv preprint physics/0505137 (2005)
7. Feng, F., He, X., Liu, Y., Nie, L., Chua, T.S.: Learning on partial-order hypergraphs. In: Proceedings of the 2018 World Wide Web Conference, WWW 2018, pp. 1523–1532. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland (2018). https://doi.org/10.1145/3178876.3186064
8. Greene, D., Sheridan, G., Smyth, B., Cunningham, P.: Aggregating content and network information to curate Twitter user lists. In: Proceedings of the 4th ACM RecSys Workshop on Recommender Systems and the Social Web, RSWeb 2012, pp. 29–36. ACM, New York (2012). https://doi.org/10.1145/2365934.2365941
9. Hadley, S.W., Mark, B.L., Vannelli, A.: An efficient eigenvector approach for finding netlist partitions. IEEE Trans. Comput.-Aided Design Integr. Circ. Syst. **11**(7), 885–892 (1992)
10. Hein, M., Setzer, S., Jost, L., Rangapuram, S.S.: The total variation on hypergraphs - learning on hypergraphs revisited. In: Proceedings of the 26th International Conference on Neural Information Processing Systems, NIPS 2013, vol. 2, pp. 2427–2435. Curran Associates Inc., USA (2013). http://dl.acm.org/citation.cfm?id=2999792.2999883
11. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J. Sci. Comput. **20**(1), 359–392 (1998). https://doi.org/10.1137/S1064827595287997
12. Karypis, G., Kumar, V.: Multilevel k-way hypergraph partitioning. VLSI Design **11**(3), 285–300 (2000)
13. Kim, S., Nowozin, S., Kohli, P., Yoo, C.D.: Higher-order correlation clustering for image segmentation. In: Advances in Neural Information Processing Systems, pp. 1530–1538 (2011)

14. Leordeanu, M., Sminchisescu, C.: Efficient hypergraph clustering. In: Proceedings of the 15th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 22, pp. 676–684. PMLR (2012). http://proceedings.mlr.press/v22/leordeanu12.html
15. Liu, H., Latecki, L.J., Yan, S.: Robust clustering as ensembles of affinity relations. In: Advances in Neural Information Processing Systems (2010)
16. Louis, A.: Hypergraph Markov operators, eigenvalues and approximation algorithms. In: Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing, STOC 2015, pp. 713–722. ACM, New York (2015)
17. Newman, M.E.: Modularity and community structure in networks. Proc. Natl. Acad. Sci. **103**(23), 8577–8582 (2006)
18. Newman, M.E.: Networks: An Introduction. Oxford University Press Inc., New York (2010)
19. Papa, D.A., Markov, I.L.: Hypergraph partitioning and clustering. In: In Approximation Algorithms and Metaheuristics. Citeseer (2007)
20. Rotá Bulo, S., Pelillo, M.: A game-theoretic approach to hypergraph clustering. IEEE Trans. Pattern Anal. Mach. Intell. **35**(6), 1312–1327 (2013)
21. Saito, S., Mandic, D., Suzuki, H.: Hypergraph p-laplacian: a differential geometry view. In: AAAI Conference on Artificial Intelligence (2018)
22. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI Mag. **29**(3), 93 (2008)
23. Shashua, A., Zass, R., Hazan, T.: Multi-way clustering using super-symmetric nonnegative tensor factorization. In: Proceedings of the 9th European Conference on Computer Vision, ECCV 2006, vol. IV, pp. 595–608. Springer, Heidelberg (2006). https://doi.org/10.1007/11744085_46
24. Yang, J., Leskovec, J.: Defining and evaluating network communities based on ground-truth. In: Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics, MDS 2012, pp. 3:1–3:8. ACM, New York (2012). https://doi.org/10.1145/2350190.2350193
25. Zhang, M., Cui, Z., Jiang, S., Chen, Y.: Beyond link prediction: predicting hyperlinks in adjacency space. In: AAAI Conference on Artificial Intelligence (2018)
26. Zhao, X., Wang, N., Shi, H., Wan, H., Huang, J., Gao, Y.: Hypergraph learning with cost interval optimization. In: AAAI Conference on Artificial Intelligence (2018)
27. Zhou, D., Huang, J., Schölkopf, B.: Learning with hypergraphs: clustering, classification, and embedding. In: Advances in Neural Information Processing Systems, pp. 1601–1608 (2007)