ALL FOR ONE, ONE FOR ALL: CONSENSUS COMMUNITY DETECTION IN NETWORKS

Mariano Tepper and Guillermo Sapiro

Department of Electrical and Computer Engineering, Duke University

ABSTRACT

Given an universe of distinct, low-level communities of a network, we aim at identifying the "meaningful" and consistent communities in this universe. We address this as the process of obtaining *consensual* community detections and formalize it as a bi-clustering problem. While most consensus algorithms only take into account pairwise relations and end up analyzing a huge matrix, our proposed characterization of the consensus problem (1) does not drop useful information, and (2) analyzes a much smaller matrix, rendering the problem tractable for large networks. We also propose a new parameterless bi-clustering algorithm, fit for the type of matrices we analyze. The approach has proven successful in a very diverse set of experiments, ranging from unifying the results of multiple community detection algorithms to finding common communities from multi-modal or noisy networks.

Index Terms— Community detection, consensus, bi-clustering.

1. INTRODUCTION

Networks are frequently used to describe many real-life scenarios were units interact with each other (e.g., see [1, 2] and references therein). A seemingly common property to many networks is the *community structure*: networks can be divided into (in general non-overlapping) groups such that intra-group connections are denser than inter-group ones. Finding and analyzing these communities sheds light on important characteristics of the networks and the data they represent. However, the best way to establish the community structure is still disputed. Addressing this is the topic of this work.

Let $G = (V, E, \psi)$ be the graph to analyze, where V is the set of n nodes, E is the set of edges, and $\psi : E \to \mathbb{R}^+$ is a weighting function on the edges (in the following we indistinguishably use the terms graph and network). Generically, we consider that a community-detection algorithm provides a set $\mathcal{C}$ of candidate communities ($\mathcal{C} \subset \mathbb{P}(V)$, where $\mathbb{P}(V)$ is the power set of V). Let us consider that we are provided with a pool (universe) $\{\mathcal{C}_k\}_{k=1}^{c}$ of c such sets. These candidates might come from:

- running different community-detection algorithms;
- running one algorithm with different parameters;
- running a community-detection algorithm on different modalities of the same data, by changing the set of edges and/or the function ψ (this case subsumes a network that changes over time);
- all of the above simultaneously.

Given the pool $\{\mathcal{C}_k\}_{k=1}^{c}$, we ask which is the partition $\mathcal{C}_k$ most representative of the actual community structure of the network G? For this one would need a criterion to select a specific partition and discard all others. This has proven a rather difficult task, where even the standard measure, modularity, has known shortcomings [3]. But why do we have to settle with selecting one solution from the pool? We argue that a better option is to combine the information of the different results into a new result.

Consensus/ensemble clustering is a well known family of techniques used in data analysis to solve this type of problems. Typically, the goal is to search for the so-called "median" (or consensus) partition, i.e., the partition that is most similar, on average, to all the input partitions. Recently, these ideas have begun to be adapted to the community detection problem [4, 5].

The most common form of consensus clustering involves creating an $n \times n$ matrix $\mathbf{B} = \frac{1}{c}\sum_{i=1}^{c} \mathbf{B}_k$, where $(\mathbf{B}_k)_{ij} = 1$ if $(\exists C \in \mathcal{C}_k)\ i, j \in C$, and 0 otherwise. There are many algorithms for analyzing $\mathbf{B}$, from simple techniques such as applying a clustering algorithm to it (e.g., k-means or hierarchical clustering), to more complex techniques. See [6] for a thorough survey of the area. An interesting approach was proposed in [7], where the authors look for a matrix $\mathbf{B}^*$ such that

$$\mathbf{B}^* = \operatorname*{argmin}_{\tilde{\mathbf{B}}} \sum_{k=1}^{c} \left\| \mathbf{B}_k - \tilde{\mathbf{B}} \right\|_F^2 = \operatorname*{argmin}_{\tilde{\mathbf{B}}} \left\| \mathbf{B} - \tilde{\mathbf{B}} \right\|_F^2 . \quad (1)$$

They solve (1) using Non-negative Matrix Factorization (NMF).

In the context of community detection, two works have explicitly addressed the consensus problem. In [5] the matrix $\mathbf{B}$ is simply thresholded, and its connected components give the final result. In [4] $\mathbf{B}$ is considered as the adjacency matrix of a new weighted network. Then the following steps are iteratively applied: (1) a unique non-deterministic algorithm is applied c times, (2) form $\mathbf{B}$ and threshold it to make it sparse, (3) stop if $\mathbf{B}$ is block diagonal, and (4) build a new network from $\mathbf{B}$ and go to (1).

Notice that the aggregation process used to build $\mathbf{B}$ involves loosing information contained in the individual matrices $\mathbf{B}_k$. In particular, only pairwise relations are conserved, while relations involving larger groups of nodes might be lost. In addition, using the average of several partitions might not be robust if some of them are of poor quality. All these methods involve working with an $n \times n$ matrix, which is highly prohibitive when the number of nodes n in the network becomes large.

Contributions. We propose a novel framework and perspective for consensus community detection by posing it as a bi-clustering problem. Our proposed approach has two main advantages: (1) all relations are conserved (instead of only keeping pairwise relations) and contribute to the consensus search, and (2) we use a much smaller matrix, rendering the problem tractable for large networks. We also propose a new *parameterless* bi-clustering algorithm, fit for the type of matrices we analyze. We stress that our goal is not finding a better optimum for the objective function of a given community detection method, but obtaining an overall good solution via consensus search.

The remainder of this work is organized as follows. In §2 we present the proposed approach. In §3 we discuss the experimental results, and finally we provide some closing remarks in §4.

Work partially supported by NSF, ONR, NGA, ARO, and NSSEFF.

Algorithm 1: Bi-clustering algorithm.

input : Preference matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$
output: Consensus communities $\{\mathcal{R}_t\}_{t=1}^{T}$ and the original communities they come from $\{\mathcal{Q}_t\}_{t=1}^{T}$.

$T \leftarrow 0$;
while $\|\mathbf{A}\|_1 > 0$ **do**
  $T \leftarrow T + 1$;
  Solve Problem (5) for $\mathbf{U}, \mathbf{V}$ with $q = 1$;
  $\mathcal{R}_T \leftarrow \{i,\ 1 \le i \le n,\ (\mathbf{U})_{i,1} \neq 0\}$;
  $\mathcal{Q}_T \leftarrow \{j,\ 1 \le j \le m,\ (\mathbf{V})_{1,j} \neq 0\}$;
  if *different communities cannot share nodes* **then**
    $(\forall i, j)\ i \in \mathcal{R}_T, 1 \le j \le m,\ \mathbf{A}_{ij} \leftarrow 0$;
  $(\forall i, j)\ 1 \le i \le n, j \in \mathcal{Q}_T,\ \mathbf{A}_{ij} \leftarrow 0$;
return $\{\mathcal{R}_t\}_{t=1}^{T}, \{\mathcal{Q}_t\}_{t=1}^{T}$

2. CONSENSUS COMMUNITY DETECTION

The input of the consensus algorithm is a pool $\{\mathcal{C}_k\}_{k=1}^{c}$ of candidates, that defines the universe of candidates $\mathcal{U} = \bigcup_{k=1}^{c} \mathcal{C}_k$. We also assign a weight $w_j \in \mathbb{R}^+$ to each community candidate $C_j \in \mathcal{U}$. From the set of nodes V and $\mathcal{U}$, we define an $n \times m$ matrix $\mathbf{A}$, whose rows and columns represent the $n = |V|$ nodes and the $m = |\mathcal{U}|$ candidates, respectively; the element $(\mathbf{A})_{ij} = w_j$ if the i-th node belongs to the j-th community, and 0 otherwise. We call $\mathbf{A}$ a preference matrix. In figs. 1 and 2 we can see two examples (white circles indicate the non-zero entries of $\mathbf{A}$).

The community weights indicate the importance assigned to each candidate and can take any form. The simplest form uses uniform weights $(\forall j)\ w_j = 1$, in which case $\mathbf{A}$ becomes a binary matrix. In this case, no prior information is used about the quality of the input community candidates. If we have such information, it can be freely incorporated in these weights.

We are interested in finding clusters in the product set $V \times \mathcal{U}$. Such a problem is known in the literature as bi-clustering [8, and references therein], and we are therefore formally connecting it here for the first time with consensus algorithms.

The main contribution of this work is therefore to address the problem of consensus community detection by bi-clustering the preference matrix $\mathbf{A}$. This provides a very intuitive rationale since, for each bi-cluster, we are jointly selecting a subset of nodes and a subset of communities such that the former belong to the latter. By directly analyzing $\mathbf{A}$ we keep all the information contained in $\mathcal{U}$ ($\mathbf{A}$ is a complete representation of $\mathcal{U}$). More classical consensus algorithms analyze an $n \times n$ matrix, while we, in contrast, work with a much smaller matrix, since for common networks $m \ll n$. Another important feature for analyzing very large networks is that each base algorithm does not need to see the complete network G. We can split the network in several (preferably overlapping) chunks, run one or more algorithms in each chunk, and let our bi-clustering algorithm perform the stitching.

Notice of course that if all of the base algorithms consistently make the same mistakes, these will be translated to the consensus solution, a characteristic common to all consensus algorithms.

2.1. Solving the bi-clustering problem

Two popular algorithms for bi-clustering are Penalized Matrix Decomposition (PMD) [9] and Sparse Singular Value Decomposition (SSVD) [10]. Both algorithms iterate two steps until some stopping criterion is met: (1) find one bi-cluster $\{\mathbf{u}, \mathbf{v}, s\}$, where $\mathbf{u} \in \mathbb{R}^n$, $\mathbf{v} \in \mathbb{R}^m$, $s \in \mathbb{R}^+$; (2) set $\mathbf{A} = \mathbf{A} - s\mathbf{u}\mathbf{v}^{\mathrm{T}}$. For step (1), they solve

$$\text{(PMD)} \quad \min_{\mathbf{u},\mathbf{v},s} \left\| \mathbf{A} - s\mathbf{u}\mathbf{v}^{\mathrm{T}} \right\|_F^2 \ \text{s.t.} \ \begin{matrix} \|\mathbf{u}\|_2 = 1, \|\mathbf{v}\|_2 = 1, \\ \|\mathbf{u}\|_1 \le c_1, \|\mathbf{v}\|_1 \le c_2, \end{matrix} \quad (2)$$

$$\text{(SSVD)} \quad \min_{\mathbf{u},\mathbf{v},s} \left\| \mathbf{A} - s\mathbf{u}\mathbf{v}^{\mathrm{T}} \right\|_F^2 + \lambda_1 \|\mathbf{u}\|_1 + \lambda_2 \|\mathbf{v}\|_1 . \quad (3)$$

Let $\mathcal{R}, \mathcal{Q}$ be the active sets of $\mathbf{u}$, $\mathbf{v}$, respectively. $\mathcal{R}, \mathcal{Q}$ act as indicators of the presence of a rank-one submatrix in $\mathbf{A}$: $\mathcal{R}$ selects rows (nodes), while $\mathcal{Q}$ selects columns (communities). This behavior makes PMD and SSVD very suitable for bi-clustering.

Correctly setting the parameters $c_1, c_2, \lambda_1, \lambda_2$ is crucial, since they determine the size of the bi-clusters (via the sparsity of $\mathbf{u}, \mathbf{v}$). PMD sets c_1, c_2 via cross-validation, while SSVD uses the Bayesian information criterion. In [11] a minimum description length criterion is used to set λ_1, λ_2 and the number of iterations for SSVD.

In this work, we propose to follow a different path for solving the bi-clustering problem at hand. Let us first notice that non-negative matrix factorization (NMF) [13] pursues a similar objective. For $1 \le q \le \min\{n, m\}$, NMF solves the problem

$$\min_{\mathbf{U} \in \mathbb{R}^{n \times q}, \mathbf{V} \in \mathbb{R}^{q \times m}} \|\mathbf{A} - \mathbf{U}\mathbf{V}\|_F^2 \quad \text{s.t.} \quad \mathbf{U}, \mathbf{V} \ge 0. \quad (4)$$

$\mathbf{A}$ is, in our application, a sparse positive matrix. Thus, the positivity constraints on $\mathbf{U}, \mathbf{V}$ have a sparsifying effect on them. We thus obtain sparse factors $\mathbf{U}, \mathbf{V}$ as in PMD and SSVD without introducing any parameters. Another consequence of the sparsity of $\mathbf{A}$ is that the Frobenius norm is not entirely well suited for analyzing it. It is more appropriate to use instead an L1 fitting term,

$$\min_{\mathbf{U},\mathbf{V}} \|\mathbf{A} - \mathbf{U}\mathbf{V}\|_1 \quad \text{s.t.} \quad \mathbf{U}, \mathbf{V} \ge 0. \quad (5)$$

With this change, we are also now aiming at obtaining a "median" type of result instead of the mean, which is completely in accordance with the objectives of consensus clustering discussed before.

Because of space limitations we omit the details on how to solve (5). Any standard NMF algorithm can be adapted to use the L1 norm; in this work, we modify the Alternating Direction Method of Multipliers in [14], that has shown good performance in practice.

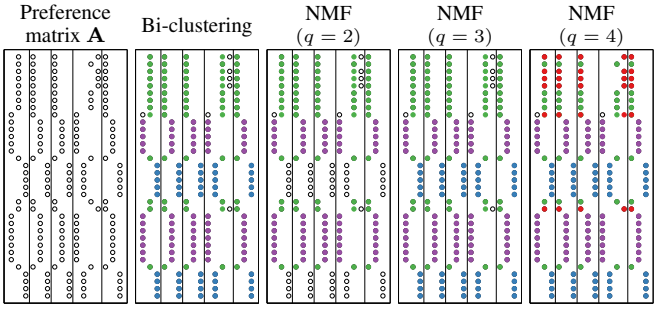


Fig. 1. The proposed iterative bi-clustering approach finds the correct number of bi-clusters (3) on the Aegean34 network [12], with 34 nodes (a different color is assigned to each pair $(\mathcal{R}_t; \mathcal{Q}_t)$, see Algorithm 1). Directly solving (5) with $q = 2$ undersegments $\mathbf{A}$ (nodes are not assigned to any community, i.e., no color, despite a lot of consistency between the base algorithms) and with $q = 4$ oversegments $\mathbf{A}$ (a single algorithm splitting a community is enough to create a new "artificial" consensus community, see the red entries).

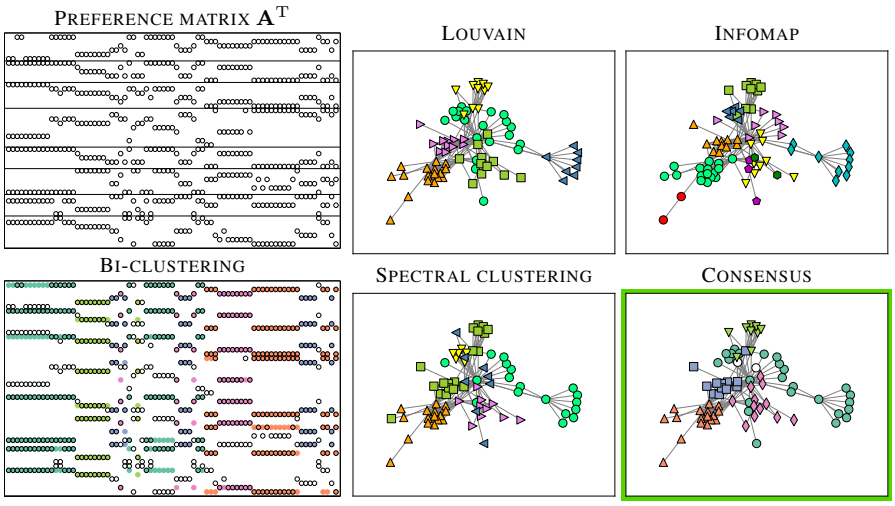
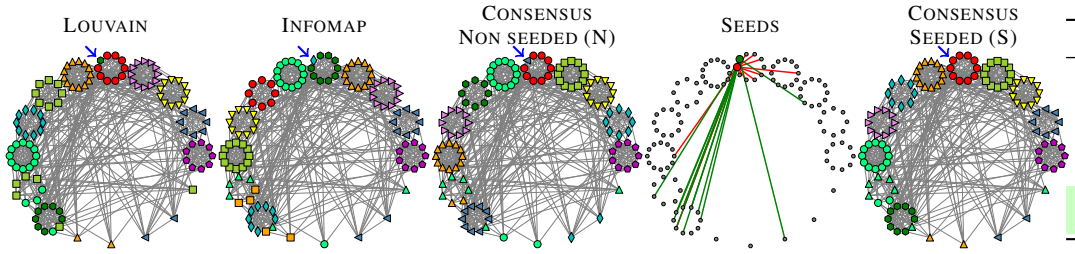


Fig. 2. Running different standard algorithms on the "Les Misérables" network [15] with 77 nodes. We show three individual results and the consensus solution. The bi-clustering result does not carbon-copy any of the individual solutions (the horizontal bands in $\mathbf{A}^{\mathrm{T}}$), but creates a new one. Also notice on the colored preference matrix (bottom left) how the algorithm "corrects" individual solutions (a different color is assigned to each pair $(\mathcal{R}_t, \mathcal{Q}_t)$, see Algorithm 1). The proposed consensus algorithm detects two nodes as singletons (in white in the bottom right graph): it is interesting to notice that the base algorithms all differ on how to treat this nodes and assign them to different communities.

Algorithm	$n = 10^2, \delta = 5$		$n = 10^2, \delta = 15$		$n = 10^3, \delta = 5$		$n = 10^3, \delta = 10$		$n = 10^4, \delta = 300$		$n = 10^4, \delta = 50$		AVG NMI
	Mod.	NMI	Mod.	NMI	Mod.	NMI	Mod.	NMI	Mod.	NMI	Mod.	NMI	
Louvain	**0.4942**	0.5646	**0.4356**	**1.0000**	**0.7571**	0.8673	**0.7591**	0.9953	**0.7646**	**1.0000**	**0.7943**	0.9995	0.9044
Infomap	0.4811	0.5142	**0.4356**	**1.0000**	0.7527	**0.8861**	0.7589	**1.0000**	**0.7646**	**1.0000**	**0.7943**	**1.0000**	0.9001
SC-(K)	0.3348	0.3334	**0.4356**	**1.0000**	0.6118	0.7823	0.6653	0.9070	0.7530	0.9553	0.7833	0.9776	0.8259
SC-($K-1$)	0.3065	0.3495	0.2855	0.6118	0.6514	0.8126	0.6624	0.8918	0.7599	0.9802	0.7846	0.9727	0.7698
SC-($K+1$)	0.4281	0.5172	0.3780	0.8514	0.6178	0.8062	0.6619	0.9103	0.7210	0.9581	0.7914	0.9686	0.8353
SC-($K+2$)	0.3611	0.3788	0.3208	0.7741	0.6006	0.7929	0.6340	0.8900	0.6988	0.9595	0.7843	0.9810	0.7961
Consensus	0.4760	**0.5840**	**0.4356**	**1.0000**	0.7501	0.8846	0.7589	**1.0000**	**0.7646**	**1.0000**	0.7937	0.9941	**0.9105**

Table 1. Results with synthetic networks (n is the number of nodes and δ the average node degree), produced with a standard benchmark generator [16]. There is no single algorithm that produces the best solution for every network; however, the consensus solution is always competitive with the best base solution (in bold). To "help" SC, K was set to the number of ground truth communities. NMI and Mod. stand for normalized mutual information and modularity, respectively.



Algorithm	Mod.	NMI
Louvain	**0.6046**	0.8964
Infomap	0.6005	0.9345
SC-(11)	0.5859	0.9054
SC-(10)	0.5928	0.8882
SC-(12)	0.5391	0.8872
Cons. (N)	0.5999	0.9309
Cons. (S)	0.5869	**0.9425**

Fig. 3. College football network [17] with 115 nodes, representing the matches between different teams. All base methods separate the node (marked with a blue arrow) from its division. By looking at its edges (in green in the 4th graph), we see that this can indeed be correct given the network. Using a little extra information, i.e., forcing the red and green nodes in the 4th graph to be in the same community, corrects this effect. As before, NMI and Mod. stand for normalized mutual information and modularity, respectively.

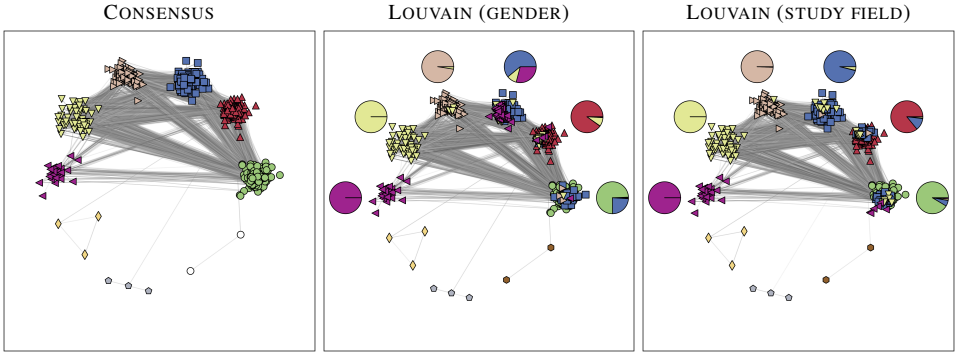


Fig. 4. Network of Facebook links between 2006 Duke graduates (part of the Facebook 100 dataset [18]) with 1424 nodes. We build two different modalities by assigning weights to the edges according to different node (student) features: field of study and gender. We find a consensus between the two different results of the Louvain algorithm. The pie charts represent the distribution of the nodes of each modality's communities with respect to the consensus.

A challenge with NMF is that q is not an easy parameter to set. To avoid a cumbersome decision process, we propose to set $q = 1$ and inscribe the L1-NMF approach in an iterative loop, as PMD and SSVD. Algorithm 1 summarizes the proposed approach. Notice that instead of subtracting $\mathbf{UV}$ from $\mathbf{A}$, we set the corresponding rows and columns to zero, enforcing disjoint active sets between the successive $\mathbf{U}_i$ and $\mathbf{V}_i$, and hence orthogonality. If the communities are allowed to share nodes, we do not change the rows of $\mathbf{A}$ (all experiments in this paper were performed with disjoint communities). An example of the effectiveness of this approach is depicted in Fig. 1.

3. EXPERIMENTAL RESULTS

For the experiments we use the following base algorithms for community detection: Louvain [19], Infomap [20], and Spectral Clustering (SC-(K)) [21], where K is the number of detected clusters/communities. For assessing the quality of a solution when ground truth is available, we use normalized mutual information (NMI). Unless specified, we use uniform weights in $\mathbf{A}$.

Several algorithms, same network. The most classical consensus scenario is when we have the result of several detection algorithms and wish to combine them into a better result.

In Fig. 2 we can observe in detail how the bi-clustering algorithm selects entries of $\mathbf{A}$ to create a new solution, preferring regularities in the matrix, while disregarding peculiarities of individual solutions ("inpainted" nodes do not have a black circle around them in the bottom left graph in Fig. 2). An important feature is that the proposed algorithm does not blindly select the best solution, but composes a consensual solution from the provided candidates.

In Table 1 we test our results using a generator of synthetic networks [16]. For this experiment, we compute the modularity $\mathrm{Mod}(\mathcal{C}_k)$ of the solution $\mathcal{C}_k$ provided by the k-th base algorithm and we use a smooth increasing nonlinear function of $\mathrm{Mod}(\mathcal{C}_k)$ as the weight for each community $C \in \mathcal{C}_k$. In general, there is no community detection algorithm that "rules them all" for every network; however our algorithm consistently performs well in all examples.

Using seeds. What happens when there is not enough information in the network to recover the "correct" structure? The College football network [17] presents a very interesting example. The network represents the matches played between teams in a season. The teams are organized in divisions, and teams should play more matches with teams from the same division than from different ones. Hence, divisions are considered as ground truth communities for this network. When we run different community detection algorithms on this network, we can observe that one of the divisions is not well recovered by any of them because one of its teams is assigned to a different community, see the blue arrows in Fig. 3. But in fact, when we observe this team's matches, it did not play against any of the teams in his division! We can add a tiny bit of a priori information by manually adding seeds to $\mathbf{A}$, i.e., by forcing some nodes (in Fig. 3 the red and green nodes in the 4th graph) to be on the same community. For this, we just modify the corresponding rows of $\mathbf{A}$ by replacing them by their disjunction (logical or). This simple seeding mechanism is able to correct the "original mistake."

One algorithm, different networks. The proposed approach also allows to combine the results of community structure algorithms that analyze different aspects of a given network (e.g., a network with different modalities or evolving over time).

The Facebook 100 dataset presents such an example. The edges represent Facebook friendship but we can also observe several node attributes (e.g., gender, major, minor, dorm, year of graduation). In our particular example, we focus on the 2006 Duke graduates. We build two modalities of this network, by assigning different weights to the edges. In the first one, we use gender information, assigning a weight of 1 if an edge links students of different sex and of 2 otherwise. In the second one, we use study field information, assigning a weight of 1 if the students do not share major nor minor, of 2 if they share major or minor, and of 3 if they share major and minor. We run the Louvain algorithm independently on these two networks and obviously obtain two different community structures, see Fig. 4. By running our consensus algorithm on these two results, we produce a solution that aggregates information from both modalities.

ρ	Best	Median	Consensus
0.1	0.9224	0.8388	**0.9258**
0.25	0.8004	0.6944	**0.8551**
0.3	0.7428	0.6315	**0.8244**
0.35	0.6801	0.5581	**0.7933**

Table 2. Results of perturbing the edge set of the US politics books network (`http://www.orgnet.com/divided.html`): $\rho|E|$ edges are exchanged at random. We provide average NMI values across 1000 trials, using 10 perturbed networks per trial. The result of Infomap in the original network is considered the ground truth. The consensus solution outperforms all individual ones, with the performance gap increasing as more edges are perturbed.

Another interesting example occurs when the network connectivity changes over time or when different modalities exhibit different edge sets. This is important for example when the graph is obtained through inference, because differences and/or errors in the inference process might yield different connectivities. We simulate such an example by taking a network and building 10 perturbed copies, randomly reassigning a subset of its edges. We then run Infomap on each copy and compare the result in terms of NMI with the Infomap result on the original network (Table 2). When a small portion of edges is perturbed, the best individual solution is still good, because there is a non-negligible chance that one of the perturbations does not alter the community structure of the original network. This is no longer true as the number of perturbed edges increases. Our algorithm is able to balance out the peculiarities of the perturbed solutions, obtaining a solution much closer to the original one and hence more resilient to perturbations.

4. CONCLUSIONS

We analyzed the process of generating consensual community detections and formalized it as a bi-clustering problem. This offers a new perspective for this important problem. Our characterization of the consensus problem offers several advantages: (1) there is no need to drop useful information, contrarily to classical consensus algorithms where only pairwise relations are considered; (2) we analyze a much smaller matrix, rendering the problem tractable for large networks. We also propose a new *parameterless* bi-clustering algorithm, fit for the type of matrices we analyze. The approach has proven successful in a very diverse set of experiments.

As future work, it would be interesting to analyze massive networks. Individual base algorithms would only analyze a portion of the network, and our technique would be used for merging the results (consensus + stitching). We finally stress that the presented concepts are general and exceed the community detection domain. They can be applied to general data clustering, extending this novel perspective to a more general scenario.

5. REFERENCES

[1] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, vol. E 69, no. 026113, 2004.

[2] R. Aldecoa and I. Marín, "Jerarca: Efficient analysis of complex networks using hierarchical clustering," *PLoS ONE*, vol. 5, no. 7, pp. e11585+, 2010.

[3] A. Lancichinetti and S. Fortunato, "Limits of modularity maximization in community detection," *Phys. Rev. E*, vol. 84, no. 6, pp. 066122+, 2011.

[4] A. Lancichinetti and S. Fortunato, "Consensus clustering in complex networks," *Sci. Rep.*, vol. 2, 2012.

[5] R. Campigotto, J. L. Guillaume, and M. Seifi, "The power of consensus: Random graphs have no communities," in *ASONAM*, 2013.

[6] S. Vega-Pons and J. Ruiz-Shulcloper, "A survey of clustering ensemble algorithms," *Int. J. of Pattern Recognit. Artif. Intell.*, vol. 25, no. 03, pp. 337–372, 2011.

[7] T. Li, C. Ding, and M. I. Jordan, "Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization," in *ICDM*, 2007.

[8] E. E. Papalexakis, N. D. Sidiropoulos, and R. Bro, "From K-Means to Higher-Way Co-Clustering: Multilinear decomposition with sparse latent factors," *IEEE Trans. Signal Process.*, vol. 61, no. 2, pp. 493–506, Jan. 2013.

[9] D. M. Witten, R. Tibshirani, and T. Hastie, "A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis," *Biostatistics*, vol. 10, no. 3, pp. 515–534, 2009.

[10] M. Lee, H. Shen, J. Z. Huang, and J. S. Marron, "Biclustering via sparse singular value decomposition," *Biometrics*, vol. 66, no. 4, pp. 1087–1095, 2010.

[11] I. Ramírez and M. Tepper, "Bi-clustering via MDL-based matrix factorization," in *CIARP*, 2013.

[12] T. S. Evans, R. J. Rivers, and C. Knappett, "Interactions in space for archaeological models," *Adv. Complex Syst.*, vol. 15, no. September, pp. 1150009+, 2011.

[13] P. Paatero and U. Tapper, "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values," *Environmetrics*, vol. 5, no. 2, pp. 111–126, 1994.

[14] Y. Xu, W. Yin, Z. Wen, and Y. Zhang, "An alternating direction algorithm for matrix completion with nonnegative factors," *Front. Math. China*, vol. 7, no. 2, pp. 365–384, 2012.

[15] D. E. Knuth, *The Stanford GraphBase: a platform for combinatorial computing*, ACM, New York, NY, USA, 1993.

[16] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Phys. Rev. E*, vol. 78, no. 4, 2008.

[17] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 99, no. 12, pp. 7821–7826, 2002.

[18] A. Traud, P. Mucha, and M. Porter, "Social structure of facebook networks," *Physica A*, vol. 391, no. 16, pp. 4165–4180, 2011.

[19] V. Blondel, J. L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech.*, vol. 2008, no. 10, pp. P10008+, 2008.

[20] M. Rosvall and C. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 105, no. 4, pp. 1118–1123, 2008.

[21] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *NIPS*, 2001.