Community Detection via Graph Optimization Methods based on Modularity and Balanced Cuts

Francesco Ferretto, Andrea Sinigaglia, Giuliano Squarcina

April 26, 2021

Contents

Abstract

The aim of this project consist into the comparisons of different methods for detecting community structures inside networks.[1]

Part I
What stands for Community Detection?

Before getting deeper we need to define the main protagonist, the *community structure*. In the study of complex networks:

a network is said to have ***community structure*** *if the nodes of the network can be easily grouped into (potentially overlapping) sets of nodes such that each set of nodes is densely connected internally.*

[1]In order to overcome misunderstandings, we'll use the notation of the correspondent paper for a specific framework.

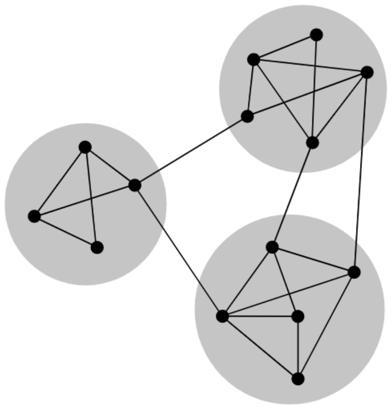

Figure 1: A sketch of a small network displaying **community structure**, with three groups of nodes with dense internal connections and sparser connections between groups.

In the particular case of *non-overlapping* community finding, this implies that the network divides naturally into groups of nodes with *dense* connections internally and *sparser* connections between groups. But *overlapping* communities are also allowed.
Finding community structures within an arbitrary network is called *Community Detection*, a task that can be a computationally difficult. The number of communities, if any, within the network is typically *unknown* and the communities are often of *unequal size* and/or *density.*

Despite these difficulties, however, several methods for community finding have been developed and employed with varying levels of success:

- **Minimum-cut method**
  One of the oldest algorithms for dividing networks into parts is the minimum cut method (and variants such as ratio cut and normalized cut). This method sees use, for example, in load balancing for parallel computing in order to minimize communication between processor nodes.

In the minimum-cut method, the network is divided into a predetermined number of parts, usually of approximately the same size, chosen such that the number of edges between groups is minimized. The method works well in many of the applications for which it was originally intended but is less than ideal for finding community structure in general networks since it will find communities regardless of whether they are implicit in the structure, and it will find only a fixed number of them.

- **Modularity maximization**

  In spite of its known drawbacks, one of the most widely used methods for community detection is modularity maximization. Modularity is a benefit function that measures the quality of a particular division of a network into communities. The modularity maximization method detects communities by searching over possible divisions of a network for one or more that have particularly high modularity. Since exhaustive search over all possible divisions is usually intractable, practical algorithms are based on approximate optimization methods such as greedy algorithms, simulated annealing, or spectral optimization, with different approaches offering different balances between speed and accuracy. A popular modularity maximization approach is the Louvain method, which iteratively optimizes local communities until global modularity can no longer be improved given perturbations to the current community state. An algorithm that utilizes the RenEEL scheme, which is an example of the Extremal Ensemble Learning (EEL) paradigm, is currently the best modularity maximizing algorithm.

  The usefulness of modularity optimization is questionable, as it has been shown that modularity optimization often fails to detect clusters smaller than some scale, depending on the size of the network (**resolution limit**); on the other hand the landscape of modularity values is characterized by a huge degeneracy of partitions with high modularity, close to the absolute maximum, which may be very different from each other. **Modularity** is one measure of the structure of networks or graphs. It was designed to measure the strength of division of a network into modules (also called groups, clusters or communities). Networks with high modularity have dense connections between the nodes within modules but sparse connections between nodes in different modules. Modularity is often used in optimization methods for detecting community structure in networks. However, it has been shown that modularity suffers a resolution limit and, therefore, it is unable to detect small communities. Biological networks, including animal brains, exhibit a high degree of modularity.

- *Statistical inference*
- *Clique-based methods*
- ...

All the methods that we'll encounter through this paper live in the mathematical setting of **Constrained Fractional Set Programs**, which consist in the optimization of a ratio of *set functions*[2], that expresses a topological feature of a graph, G.

Part II
Modularity Based Approach

Fast-Atvo

Maximizing the modularity set function Q of a network as a function of a set of nodes S (a.k.a. *leading community problem*) is a NP-complete combinatorial optimization problem:

$$\max_{S\subseteq V} Q(S) = \frac{1}{vol(G)} \sum_{i,j\in S} \left(A_{ij} - \frac{d_i d_j}{vol(G)} \right), \quad \text{with:}$$

- $G = (V, E)$ an undirected graph with node set $V = \{1, \dots, n\}$ and edge set E; A a non negative weight matrix such that $A_{ij} > 0$ if and only if $ij \in E$;
- d the vector of degrees such that $d_i = \sum_j A_{ij}$;
- $vol(G) = \sum_{i=1}^{n} d_i$ the graph volume.

The leading community problem is equivalent to the problem of maximizing a continuous real-valued function TV_Q (*modularity total variation*) subject to an arbitrary box constraint of the type $-a \le x_i \le b, \quad a, b > 0$, allowing to tackle the leading module problem with techniques from continuous optimization. In the following are provided the definitions and the results (without

[2]A *set function* is a function whose input is a set. In our case the set will be the power set of the vertex set V associated with the graph.

proof for brevity reasons) necessary to prove the equivalence. Given any graph G and any $x \in \mathrm{R}^n$ the *total variation* of G is defined:

$$TV_G(x) = \frac{1}{2}\sum_{ij} A_{ij}|x_i - x_j|$$

The Lovász extension f_Q of Q is defined as:

$$f_Q(x) = \sum_{i=1}^{n-1} Q(C_{i+1})(x_{i+1} - x_i) + Q(V)x_1$$

The *cut function* K is defined as:

$$K(S) = \sum_{i\in S, j\notin S} A_{ij}$$

so $K(S)$ is the cut function using the *actual* weights of edges. For any graph G and any $x \in \mathrm{R}^n$ the relation between the *Lovász extension* of the *cut function* and the $TV_G(x)$ is:

$$f_K(x) = TV_G(x)$$

Moreover $Q(S)$ can be expressed in terms of *cut functions*:

$$Q(S) = \frac{1}{vol(G)}\Big(K_0(S) - K(S)\Big)$$

where $K_0(S) = \sum_{i\in S, j\notin S_{ij}S} d_i d_j/vol(G)$ (expected weights of edges under random graph G_0). Applying the Lovász extension to the above expression:

$$\begin{aligned}
f_Q(x) &= f_{\{\frac{1}{vol(G)}(K_0(S)-K(S))\}}(x)\\
&= \frac{1}{vol(G)} f_{\{\sum_{i\in S, j\notin S_{ij}} d_i d_j/vol(G) - \sum_{i\in S, j\notin S_{ij}} A_{ij}\}}(x)\\
&= \frac{1}{vol(G)} f_{\{\sum_{i\in S, j\notin S_{ij}} (d_i d_j/vol(G) - A_{ij})\}}(x)\\
&= \frac{1}{vol(G)} f_{\{\sum_{i\in S, j\notin S_{ij}} M\}}(x)\\
&= \frac{1}{vol(G)} f_{\{K_M(S)\}}(x)\\
&= \frac{1}{vol(G)} \cdot \frac{1}{2}\sum_{ij}\left(\frac{d_i d_j}{vol(G)} - A_{ij}\right)|x_i - x_j|\\
&= \frac{1}{vol(G)} TV_Q(x)
\end{aligned}$$

So the Lovász extension of Q is the total variation on a graph with real valued adjacency matrix $M = (d_i d_j/vol(G) - A_{ij})$, i.e., the graph whose nodes are V and such that a weighted signed edge with weight M_{ij} exists between i and j if and only if $M_{ij} \neq 0$. Moreover the following identity holds:

$$Q(S) = \frac{TV_Q(u_S)}{(a+b)\cdot vol(G)}$$

for $u_S = b1_S - a1_{\bar{S}}$. All the previous definitions and results allow to claim:

$$\max_{S\subseteq V} Q(S) = \max_{-a\le x_i\le b} \frac{TV_Q(x)}{vol(G)\cdot(a+b)}$$

Thus maximizing Q is equivalent (up to scaling) to maximize TV_Q over the box $\mathcal{B}(a,b) = \{x \in \mathrm{R}^n : -a \le x_i \le b, i = 1,\dots,n\}$. Note that the maximum x^* is always attained on the border of $\mathcal{B}(a,b)$, so it is of the form $x^* = b1_S - a1_{\bar{S}}$.
In order to optimize TV_Q, which is continuous but non-smooth, an approximation $\widetilde{TV_Q} = TV_Q^p$ with continuous derivatives (so smooth) is computed:

$$\begin{aligned}
TV_Q^p(x) &= \frac{1}{2}\sum_{ij}\left(\frac{d_i d_j}{vol(G)} - A_{ij}\right)|x_i - x_j|^p\\
&= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} M_{ij}|x_i - x_j|^p
\end{aligned}$$

From the second equality is evident that the function evaluation (and so its gradient) requires a number of arithmetic operations quadratic in n:

$$\mathcal{O}\left(\frac{n(n-1)}{2}\right)$$

so some tricks are used to reduce this cost (more details in next subsection). The problem can be restated as:

$$\min_{-a\le x_i\le b} -TV_Q^p(x)$$

Also for this problem good solutions are likely to lie on the boundary of $\mathcal{B}(a,b)$, allowing to select at each iteration a set of variables (*Active set*) to be fixed at the boundary and a set of variables (*Working set*) to optimize. Moreover the use of a *non-monotone stabilization technique* avoids the computation at every iteration of the objective function TV_Q^p (which is more expensive to compute than Q). This approach is named *Fast Active-SeT based Approximate Total Variation Optimization* algorithm (`FAST-ATVO`). Additionally a *global optimization strategy* called *Partition and Swap* (`PS`) perturbates the current x^* in order to increase TV_Q^p. More in detail, the procedure is described in the following subsections.

FAST-ATVO

Let $x \in \mathrm{R}^n$ be an initial solution obtained as the *leading eigenvector* of the modularity matrix M with entry ij:

$$M_{ij} = A_{ij} - \frac{d_i d_j}{vol(G)} \quad \forall\, i, j \in V$$

and then shifted on the boundary of $\mathcal{B}(a, b)$ following the criterium:

$$\begin{cases} x_i = -a & \text{if} \quad x(i) < 0 \\ x_i = b & \text{if} \quad x(i) \geq 0 \end{cases}$$

This is justified by:

1. the relation between the *Rayleigh quotient* r_M of modularity matrix M and the modularity $Q(S)$ (see [10] and [5]):

   $$\frac{\langle \upsilon_S, M\, \upsilon_S \rangle}{\|\upsilon_S\|^2} = 4 \cdot Q(S) \implies$$
   $$\max_{x \in \{-1,1\}} \frac{1}{4} \cdot r_M = \max_{S \subseteq V} Q(S) \approx \max_{x \in \mathrm{R}^n} \frac{1}{4} \cdot r_M$$

   with $\upsilon_S = 1_S - 1_{\bar{S}}$ and choosing the weight w $\forall$ node i such that: $\sum_{i=1}^n w_i = vol(G) = \|\upsilon_S\|^2$. Exploiting the fact that $\max\{r_M\} = \lambda^*(M)$, where $\lambda^*(M)$ indicate the biggest eigenvalue of matrix M, then the optimal set S can be found using the spectral decomposition of M from linear algebra.

2. the objective function $TV_Q^p(x)$ is likely to have optimal value on boundaries, so force the initial solution to be there.

A `while` cycle is performed until x^* that minimizes the objective function $f(x)$ is found. To do so, the *stationarity* condition:

$$\|x - [x - \nabla f(x)]^{\#}\,\| = 0$$

with $[x - \nabla f(x)]^{\#} = \max(-a, \min(x - \nabla f(x), b))$ the projection on feasible set, has to be satisfied. Additionally $f(x^*)$ has to be smaller than minimum value of $f(x)$ found until now. However the above condition cannot guarantee convergence to a global minimum, but just to a stationary point (even if in practice it provides often a good result). This drawback will be reduced using:

- randomized choice of working set W
- *non-monotone linesearch*
- `PS` strategy (see next subsection)

Since the computational cost of $f(x) = TV_Q^p(x)$ is quadratic, an alternative formulation is used:

$$f(x) = \frac{\nabla f(x)^T\, x}{p}$$

It allows a linear cost because $\nabla f(x)$ has to be computed at every iteration (to check stationarity), and when $\nabla f(x)$ is known, then the new $f(x)$ has a computational cost $\mathcal{O}(n)$. The only drawback is that, if the unit stepsize is not accepted, the $f(x)$ (and so $\nabla f(x)$) could be computed many times before producing the successive x. Usually this is an unlikely event.
The *Active set A* is computed as:

$$\begin{aligned} A(x) = \{ i : \, x_i = l_i \quad &\wedge \quad \nabla_i f(x) > 0 \\ &\vee \\ i : \, x_i = u_i \quad &\wedge \quad \nabla_i f(x) < 0 \} \end{aligned}$$

and its complement is the *Non-Active set N*. The probable location of x^* on the boundary is exploited to save computational resources by using only a random sampled subset of variables from N for optimization, called the *Working set W*. The $|W|$ is initially small and is increased at each iteration to increase the chance to find x^*, until a *max* size is reached, otherwise the benefit of using W would be completely lost after many iterations. Since only variables in W are updated, to speed up the convergence is useful to include in W the variable $x_{\hat{\imath}}$ with index $\hat{\imath}$ that most violates stationarity:

$$\max_i \left\{ x_i - \max(-a, \min(x - \nabla f(x), b)) \right\}$$

Further computational time is saved evaluating the $f(x)$ every Z iterations. If $f(x) \geq f_{max}$, with f_{max} the maximum among the last m function evaluations, then x is the worst one found in last m evaluations, so it's necessary to backtrack to x^* found until now. If $f(x^*) < f(x) < f_{max}$ insert $f(x)$ in last m function evaluations and recompute f_{max}. If $f(x) \leq f(x^*)$ then update the value of x^* and $f(x^*)$.
The search direction d_W is computed:

$$d_W = -\frac{1}{\mu^k} \nabla_W f(x)$$

with μ^k computed as in Barzilai−Borwein (see [**barzilai**]) and only for the variables in W, while all the others are equal to zero. To guarantee the global convergence of the algorithm is crucial the following result:

$$\lim_{k \to \infty} \nabla f(x^k)^T\, d_W^k = 0$$

with k the iteration number. In a coding implementation, since an exact satisfaction is impossible/inefficient, convergence is assumed when $\nabla f(x)^T d_W < gd_{min}$ with $gd_{min} < 0$ a suitable threshold. This is because to ensure a sufficient decrease of the objective function, the quantity $\nabla f(x)^T d_W$ has to be enough negative. Otherwise the value of $f(x)$ should be computed, and if $f(x) \leq f(x^*)$ the cycle should finish since convergence has been reached.
Then, if $\| [x + d_W]^{\#} \|$ is sufficiently small, x is updated to $[x + d_W]^{\#}$ and $\nabla f(x)$ is updated following two different formulas depending on $|W|$:

- if $|W| \geq \frac{n-1}{3}$:

$$\nabla_i f(x) = p \sum_{\substack{j=1 \\ j \neq i}}^{n} M_{ij} sign(x_i - x_j)|x_i - x_j|^{p-1}$$

- else:

$$\nabla_i f(x) = \begin{cases} \phi_i(x) + \varrho_i(x) & i \in W \\ \phi_i(x) + \nabla_i f(x^{old}) - \phi_i(x^{old}) & i \notin W \end{cases}$$

with:

$$\phi_i(x) = p \sum_{\substack{j=1 \\ j \neq i \\ j \in W}}^{n} M_{ij} sign(x_i - x_j)|x_i - x_j|^{p-1}$$

$$\varrho_i(x) = p \sum_{\substack{j=1 \\ j \neq i \\ j \notin W}}^{n} M_{ij} sign(x_i - x_j)|x_i - x_j|^{p-1}$$

The current iteration is terminated without computing the Armijo linesearch. This means that the stepsize $\delta^\nu = 1$ (unit stepsize), since $\nu = 0$.
Otherwise, if $f(x) \geq f_{max}$, it's necessary to backtrack to x^*; if $f(x^*) < f(x) < f_{max}$ insert $f(x)$ in last m function evaluations and recompute f_{max}. If $f(x) \leq f(x^*)$ then update the value of x^* and $f(x^*)$. In all last 3 cases *non-monotone* Armijo linesearch has to be computed.
Classic Armijo linesearch guarantees a monotonical decrease of the objective function, but enforcing monotonicity of the function values by selecting a stepsize non-unitary can considerably slow the rate of convergence. The *non-monotone* Armijo linesearch [7] allows a faster rate of convergence imposing that the function value of each new iterate is lower than the maximum function value of a prefixed number of previous iterates (so it could be higher than the function value at the previous iterate, hence is not monotonically decreasing). The stepsize δ^ν is choosen such that ν is the smallest non-negative integer satisfying:

$$f([x + \delta^\nu d]^{\#}) \leq f_{max} + \gamma \delta^\nu \nabla f(x)^T d$$

with $\gamma = 10^{-3}$ as suggested in [7].
Then $|W|$ is increased by one if and only if it is below the *max size of* W:

$$|W| = \min(|W| + 1, |W|_{max})$$

Then the cycle restarts.

Note on convergence

The *stationarity* condition:

$$\|x - [x - \nabla f(x)]^{\#} \| = 0$$

is equivalent to:

$$\begin{aligned} \nabla_i f(x) = 0, &\quad i : -a < x_i < b \\ \nabla_i f(x) \geq 0, &\quad i : x_i = -a \\ \nabla_i f(x) \leq 0, &\quad i : x_i = b \end{aligned}$$

which can be expressed in compact notation as:

$$\phi(x_i) = 0, \quad i = 1, \ldots, n$$

with

$$\phi(x_i) = \min\Big\{ \max\{-a - x_i, -\nabla_i f(x)\}^2, \max\{x_i - b, \nabla_i f(x)\}^2 \Big\}$$

Note that:

- $\phi(x_i) \geq 0 \quad \forall i$
- $\phi(x_i) = 0 \quad if \quad \nabla_i f(x) = 0$

By contradiction, assume that x is *non-stationary*. Then exists at least an index i such that $\phi(x_i) > 0$ and so $\nabla_i f(x) \neq 0$. Hence has to exist $\varepsilon \in (0, 1)$ sufficiently small such that $\phi(x_i) \geq \varepsilon \quad \forall k \geq \hat{k}$. The index i can belong to:

► *Active set A*:
this implis $x_i = -a$ or $x_i = b$. Moreover for k big enough $\nabla_i f(x) \geq -\frac{\varepsilon}{2}$. Computing $\phi(x_i)$ using these values of x_i and $\nabla_i f(x)$ leads to:

$$\phi(x_i) \leq \frac{\varepsilon^2}{4}$$

which is $< \varepsilon$. So $\nexists \varepsilon > 0 : \quad \varepsilon \leq \phi(x_i), \implies \phi(x_i) = 0$.

▶ *Working set W*:
exploiting the result:

$$\lim_{k\to\infty} \nabla f(x^k)^T\, d_W^k = 0$$

and the definition of d is possible to write:

$$\begin{aligned}\lim_{k\to\infty} \nabla f(x^k)^T\, d_W^k &= \lim_{k\to\infty} \nabla f(x^k)^T \left(-\frac{1}{\mu}\nabla f(x^k)\right)\\ &= \lim_{k\to\infty} -\frac{1}{\mu}\cdot \|\nabla f(x^k)\|^2\\ &= 0 \quad\Longleftrightarrow\quad \nabla_i f(x) = 0\\ &\Longrightarrow\ \phi(x_i) = 0\end{aligned}$$

▶ $N \setminus W$:
Assume that index $\hat{\imath}$ (the one which most violates stationarity) is constant for all k. So it's true that:

$$\begin{aligned}|x_i^k - [x^k - \nabla f(x^k)]_i^{\#}| &\le |x_{\hat{\imath}^k}^k - [x^k - \nabla f(x^k)]_{\hat{\imath}^k}^{\#}|\\ &= |x_{\hat{\imath}}^k - [x^k - \nabla f(x^k)]_{\hat{\imath}}^{\#}|\end{aligned}$$

Since W was built including index $\hat{\imath}$, $\hat{\imath} \subseteq W$ and so, by previous point, $\phi(x_{\hat{\imath}}) = 0$. If the point $x_{\hat{\imath}}$ that most violates stationarity is actually stationarity, then any other point x_i is stationarity. Hence $\phi(x_i) = 0$.

The initial hypothesis is always false, then x is *stationary*.

PS

In order to improve the quality of the final solution, the following procedure is applied for a predetermined number of times:

1) a given percentage σ of variables is moved to the bound of $\mathcal{B}(a, b)$ with the opposite sign, i.e. if $x_i \le 0$ then its value is changed to be equal to b. So the variables selected are swapped to the opposite boundary. The variables $x_i = 0$ are assigned with same probability to one of the boundaries. Call them x^{swap}.

2) FAST-ATVO is runned using as starting point x^{swap}, obtaining a new final solution f^{new}. The corresponding modularity Q^{new} is computed and if:

$$Q^{new} > Q^*$$

then the optimal solution is updated, $x^* = x^{new}$ and $Q^* = Q^{new}$.

Part III
Ratio DCA

Another approach for *Graph Clustering* (a.k.a. Community Detection), in a constrained fashion[3], is to set an explicit optimization problem. From the previous literature, other authors focused their attention on the *poly*-time ($O(n)$) execution of the algorithms' requisite, contrarily to Bühler et al.[3] who followed the solving strategy adopted by Mahoney et al.[8], writing an *explicit optimization problem*, which is based onto the *constrained optimization of a ratio of set functions*, whose general mathematical formulation is the following:

$$\min_{C\subseteq V} \frac{\widehat{R}(C)}{\widehat{S}(C)} =: \widehat{Q}(C) \quad (1)$$

$$\text{subject to } : \widehat{M_i}(C) \le k_i, \quad i = 1, \ldots, K$$

where:

- $\widehat{R}, \widehat{S}, \widehat{M_i} : \{0,1\}^{|V|} \to \mathbb{R}$ are set functions on a set $V = \{1, \ldots, n\}$;
- We assume here that $\widehat{R}, \widehat{S}$ are *nonnegative* and that $\widehat{R}(\emptyset) = \widehat{S}(\emptyset) = 0.$;
- the constraint functions $\widehat{M_i}$ are *not* required to be non-negative;

Mathematical notation that will use to refer to the graph:

- $G = (V, W)$ denotes an *undirected*, *weighted graph* with a non-negative, symmetric weight matrix $W \in \mathrm{R}^{n\times n}$, where $n = |V|$;
- Given $C \subseteq V$, then its generalized volume is defined as $vol_g(C) = \sum_{i\in C} g_i$, in particular, if:

$$g_i = \begin{cases} 1 & \mathrm{vol}_g(C) = |C| \\ d_i & \mathrm{vol}_g(C) = vol_d(C) \end{cases}$$

where the degree of a vertex i is

$$d_i = \sum_{j\in V} w_{ij}.$$

[3] *Constrained Graph clustering* is a *semisupervised* approach to clustering data while incorporating domain knowledge in the form of constraints. The constraints are usually expressed as pairwise statements indicating that two items must, or cannot, be placed into the same cluster. Constrained clustering algorithms may enforce every constraint in the solution, or they may use the constraints as guidance rather than hard requirements.

- $\bar{A} = V \backslash A$ denotes the complement of A.

Respect to the modularity based approach, that still enter in the framework of constrained fractional set programs, the following methods are *locally optimal.*

Constrained Balanced Graph Cuts for Local Clustering and Constrained Local Community Detection

The general setting of the latter mathematical optimization problem incorporates two formulations of interest for our intent. One approach is the *Constrained Maximum Density* and the other is the *Constrained Balanced Graph Cut*, that are formulated as follows.

Constrained (Local) Graph Clustering via Minumum Normalized Cut

The general local clustering problem can then be formulated as

$$\min_{C \subseteq V} \frac{\mathrm{cut}(C, \bar{C})}{\widehat{S}(C)} \quad (2)$$

$$\text{subject to : } \mathrm{vol}_g(C) \leq k, \text{ and } J \subseteq C$$

where:

- $\mathrm{cut}(C, \bar{C}) := \sum_{i \in C, j \in \bar{C}} w_{ij}$
- $\mathrm{vol}_g(C) = \sum_{i \in C} g_i$ represents the *general volume* of C,
- Let $\widehat{S}$ be a symmetric balancing function, $\widehat{S}(C) = \mathrm{vol}_d(C)\,\mathrm{vol}_d(\bar{C})$ for the normalized cut,
- $J \subseteq C$ denotes the *seed set*,
- g and $h \in \mathbb{R}^n_+$ are the *vertex weights*,
- k is the upper bound for $\mathrm{vol}_h(C)$.

The choice of the balancing function $\widehat{S}$ allows the user to influence the trade-off between getting a partition with small cut and a balanced partition.

In order to compare to the method of Mahoney et al.[8], we restrict ourselves in this report to the normalized cut with volume constraints, that is $\widehat{S}(C) = \mathrm{vol}_d(C)\,\mathrm{vol}_d(\bar{C})$ and $g = d$

$$\min_{C \subseteq V} \frac{\mathrm{cut}(C, \bar{C})}{\mathrm{vol}_d(C)\,\mathrm{vol}_d(\bar{C})} \quad (3)$$

$$\text{subject to : } \mathrm{vol}_d(C) \leq k, \text{ and } J \subseteq C$$

Notice that:

$$\begin{aligned}\mathrm{NCut}(C) &= \frac{cut(C, \bar{C})}{vol_d(C) \cdot vol_d(\bar{C})} \\ &= \frac{\sum_{i \in C} \sum_{j \in \bar{C}} w_{ij}}{(\sum_{i \in C} d_i) \cdot (\sum_{i \in \bar{C}} d_i)} \\ &= \frac{\sum_{i \in C} \sum_{j \in \bar{C}} w_{ij}}{(\sum_{i \in C} \sum_{j \in V} w_{ij}) \cdot (\sum_{i \in \bar{C}} \sum_{j \in V} w_{ij})} \leq 1\end{aligned}$$

Moreover since it is a ratio of non negative quantities, $\mathrm{Ncut}(C) \geq 0$.

Constrained (Local) Community Detection via Maximum Density

In *community detection*, one approach is to find an highly connected set, i.e. to look for a set C which has high *association*, defined as

$$assoc(C) = \sum_{i,j \in C} w_{ij}.$$

By dividing the association of C by its size we obtain its *density*, leading to the following task:

$$\max_{C \subseteq V} \frac{\mathrm{assoc}(C)}{\mathrm{vol}_g(C)} \quad (4)$$

$$\text{subject to } : k_1 \leq \mathrm{vol}_h(C) \leq k_2, \text{ and } J \subseteq C$$

where:

- k_1, k_2 are the lower and upper bound respectively for $\mathrm{vol}_h(C)$.

Notice that:

$$\begin{aligned}\text{density(C)} &= \frac{assoc(C)}{vol_d(C)} \\ &= \frac{\sum_{i \in C} \sum_{j \in C} w_{ij}}{\sum_{i \in C} d_i} \\ &= \frac{\sum_{i \in C} \sum_{j \in C} w_{ij}}{\sum_{i \in C} \sum_{j \in V} w_{ij}} \leq 1\end{aligned}$$

Moreover since it is a ratio of non negative quantities, $\mathrm{density}(C) \geq 0$.

In the implementation we set $g_i = 1$ for the volume function vol_g and $h_i = 1$ in order to define a constraint on the cardinality.

The above constrained problem allows to identify a dense community C, containing the seed set J, that can be used to analyze the given community structure in a given graph G.

In literature has emerged that without the volume constraints the obtained communities are typically either *too large or too small.* The other reason why a seed constraint is included in the formulation is that trying to optimize the function in a neighborhood of a specific set reduces the dimensionality of the problem and at the same time allows us to incorporate some prior knowledge in the problem.

A common ground

The two problems are equivalent in a specific case. In particular when $\mathrm{vol}_g(C) = \mathrm{vol}_d(C)$, decomposing the objective of (4) analogously to the argument for the normalized cut as

$$\frac{\mathrm{assoc}(C)}{\mathrm{vol}_d(C)} = 1 - \frac{\mathrm{cut}(C, \bar{C})}{\mathrm{vol}_d(C)}$$

This implies that for:

- $\mathrm{vol}_g(C) = \mathrm{vol}_d(C)$ in (4)
- $\widehat{S}(C) = \mathrm{vol}_d(C)$ in (2)

the problem (4) *is equivalent to* (2) if we choose the same constraints. For the sake of clarity, we will present the method via *Maximum Density*, emphasizing the mathematical differences wrt the *Normalized Cut.*

In their work, Bühler et al.[3] showed that every *constrained non-negative fractional set programs* have an ***equivalent*** *tight continuous relaxation.* The continuous relaxation is possible thanks to an hungarian mathematician, László Lovász.

Lovász Extension

The crucial passage in this report is the transition from the *combinatorial optimization problem* to its *continuous counterpart*: the *Lovász Extension*, allowing us to extend a function whose domain coincides with the vertices of the hypercube in n dimension $\{0,1\}^{|V|}$ to the whole hypercube $[0,1]^{|V|}$.
Before introducing the function we need another mathematical ingredient, which compacts the extension in one single formula.

Let $f \in \mathbb{R}^n$ be the continuous representation of a subset of vertices in the hypercube $[0,1]^n$. We assume wlog that f is ordered in ascending order $f_1 \leq f_2 \leq \cdots \leq f_n$ (this is for the sake of notation's compactness). We introduce the definition of a discrete subset via continuous coordinates:

$$C_i := \{j \in V \mid f_j \geq f_i\}, \quad i = 1, \ldots, n \quad (5)$$

Given a set function $\widehat{R} : \{0,1\}^{|V|} \to \mathbb{R}$, its *Lovász Extension*[4] $R : \mathbb{R}^{|V|} \to \mathbb{R}$ is defined as:

$$R(f) = \sum_{i=1}^{n-1} \widehat{R}\left(C_{i+1}\right)\left(f_{i+1} - f_i\right) + \widehat{R}(V) f_1$$

Note that $R\left(\mathbf{1}_C\right) = \widehat{R}(C)$[5] for all $C \subseteq V$, i.e. R is indeed an extension of $\widehat{R}$ from $\{0,1\}^{|V|}$ to $\mathbb{R}^n$. By the $R\left(\mathbf{1}_C\right) = \widehat{R}(C)$ equivalence, we can observe that the *unconstrained version* of (1) has the following relaxation:

$$\inf_{f \in \mathbb{R}^n_+} \frac{R(f)}{S(f)}$$

It can be proven that the above relaxation is *tight*, meaning that the continuous and the combinatorial optimization problem are equivalent in the sense that the optimal values *agree* and the *optimal solution of the combinatorial problem can be obtained from the continuous solution.*
This result can be achieved thanks to the nature of the chosen set function(s)[6], that belongs to the class of *submodular functions*, defined as follows:

A set function $\widehat{R} : \{0,1\}^{|V|} \to \mathbb{R}$ is ***submodular*** *if for all $A, B \subseteq V, \widehat{R}(A \cup B) + \widehat{R}(A \cap B) \leq \widehat{R}(A) + \widehat{R}(B)$. It is supermodular, if the converse inequality holds true, and modular if we have equality.*

The connection between submodular set functions and convex functions is as follows:

Let $R : \mathbb{R}^{|V|} \to \mathbb{R}$ be the Lovász extension of $\widehat{R} : \{0,1\}^{|V|} \to \mathbb{R}$. Then, $\widehat{R}$ is submodular if and only if R is **convex**. *Furthermore, if $\widehat{R}$ is submodular, then* $\min_{A \subseteq V} \widehat{R}(A) = \min_{f \in [0,1]^n} R(f)$

Thus submodular minimization problems reduce to convex minimization problems.

[4]In the following, we always use the hat-symbol ($\widehat{\ }$) to denote set functions and omit it for the corresponding Lovász extension.

[5]We use $\mathbf{1}_C \in \mathbb{R}^n$ to denote the indicator vector of the set C, i.e. the vector which is 1 at entry j if $j \in C$ and 0 otherwise.

[6]density(C) and Ncut(C)

A similar equivalence of continuous and combinatorial optimization problems is the main topic of this report, but before presenting it in this context we need to list some useful properties of the Lovász extension:

Let $R : \mathbb{R}^{|V|} \rightarrow \mathbb{R}$ be the Lovász extension of $\widehat{R} : \{0,1\}^{|V|} \rightarrow \mathbb{R}$. Then,

- *R is* ***positively one-homogeneous***[7]
- *$\boldsymbol{\alpha}(f) \geq 0, \forall f \in \mathbb{R}^{|V|}$ and $R(\mathbf{1}) = 0$ if and only if $\widehat{R}(A) \geq 0, \forall A \subseteq V$ and $\widehat{R}(V) = 0$*
- *Let $S : \mathbb{R}^{|V|} \rightarrow \mathbb{R}$ be the Lovász extension of $\widehat{S} : \{0,1\}^{|V|} \rightarrow \mathbb{R}$. Then, $\lambda_1 R + \lambda_2 S$ is the Lovász extension of $\lambda_1 \widehat{R} + \lambda_2 \widehat{S}$, for all $\lambda_1, \lambda_2 \in \mathbb{R}$*

Then the core for the discussed methods of this part of the report is here presented.

Let $\widehat{R}, \widehat{S} : \{0,1\}^{|V|} \rightarrow \mathbb{R}$ be non-negative set functions and $\widehat{R} := \widehat{R}_1 - \widehat{R}_2$ and $\widehat{S} := \widehat{S}_1 - \widehat{S}_2$ be decompositions into differences of submodular set functions.
Let the Lovasz extensions of $\widehat{R}_1, \widehat{S}_2$ be given by R_1, S_2 and let R_2', S_1' be positively one-homogeneous convex functions with $S_1'(\mathbf{1}_A) = \widehat{S}_1(A)$ and $R_2'(\mathbf{1}_A) = \widehat{R}_2(A)$ such that $S_1' - S_2$ is non-negative.
Define $R := R_1 - R_2'$ and $S := S_1' - S_2$.

Then

$$\inf_{C \subseteq V} \frac{\widehat{R}(C)}{\widehat{S}(C)} = \inf_{f \in \mathbb{R}_+^{|V|}} \frac{R(f)}{S(f)}$$

Moreover, it holds for all $f \in \mathbb{R}_+^n$,

$$\frac{R(f)}{S(f)} \geq \min_{i=1,\ldots,n} \frac{\widehat{R}(C_i)}{\widehat{S}(C_i)}.$$

Let furthermore $\widehat{R}(V) = \widehat{S}(V) = 0$, then all the above statements hold if one replaces $\mathbb{R}_+^n$ with $\mathbb{R}^n$.

This result, sustained by the fact that such decomposition always exist, extends the class of functions that maintain the property of being a tight relaxation of the associated combinatorial fractional set program from symmetric non-negative set functions (whose numerator is restricted to be submodular) to arbitrary ratios of non-negative ones.
An incentive factor who led to the study of the properties of decompositions of set functions into differences was the difficulty to derive and/or work with explicit forms of the Lovasz extensions of $\widehat{R}$ and $\widehat{S}$. The above theorem shows that, given a decomposition of $\widehat{R}$ and $\widehat{S}$ into a difference of submodular set functions, one needs the Lovasz extension only for the first term of $\widehat{R}$ and the second term of $\widehat{S}$. The remaining terms can be replaced by any convex one-homogeneous functions that also extend the corresponding set functions.

[7] R: $\mathbb{R}^{|V|} \rightarrow \mathbb{R}$ is positively one-homogeneous if $R(\alpha f) = \alpha R(f), \forall \alpha \in \mathbb{R}$ with $\alpha \geq 0$

Retrieval of the discrete solution

Given a vector $f \in \mathbb{R}^n$ for the continuous problem, it is possible to construct a set C' by computing

$$C' = \underset{C_i, i=1,\ldots,n}{\arg\min} \frac{\widehat{R}(C_i)}{\widehat{S}(C_i)}$$

This particular process is defined as ***optimal thresholding*** [3].

Elimination of the volume constraint

In order to derive a tight relaxation of the leading problem, first integrate the volume constraints via a penalty term γ in order to control the trade-off between satisfying all constraints and achieving an optimal objective value. Yielding to the following:

$$\min_{C \subseteq V} \frac{\mathrm{vol_g}(C) + \gamma \widehat{T}_{k1,k2}(C)}{\mathrm{assoc}(C)}$$
$$\text{subject to} : J \subseteq C$$

where:

- $\widehat{T}_{k1,k2}(C) = \widehat{T}_{k1}(C) + \widehat{T}_{k2}(C)$ with $\widehat{T}_k(C) = \max\{0, \mathrm{vol}_h(C) - k\}$ for a specific k.

Elimination of seed constraint

Given the input graph G, the algorithm aims to find a set of nodes C containing the seed set J, so in order to achieve this task the idea is to restrict our search locally around the set J by setting a maximum distance from it, as explained in [3]. The above formulation holds for any γ such that $\gamma \geq \frac{\mathrm{vol}_g(C_0)\mathrm{vol}(V)}{\theta \mathrm{assoc}(C_0)}$ for a feasible set $C_0 \in V$.

By writing $C = A \cup J$ for some set $A \subseteq V$ with $A \cap J = \emptyset$ the problem can be reduced in finding the set A on the graph made up of all the vertices

$V\backslash J$ and therefore it's possible to incorporate the seed constraint obtaining the following equivalent problem:

$$\min_{A\subseteq V\backslash J} \frac{\mathrm{vol}_g(A)+\mathrm{vol}_g(J)+\gamma\widehat{T}_{k_1',k_2'}(A)}{\mathrm{assoc}(A)+\mathrm{assoc}(J)+2\,\mathrm{cut}(J,A)}$$

where $k_1' = k_1 - \mathrm{vol}_h(J)$ and $k_2' = k_2 - \mathrm{vol}_h(J)$. Our problem now is reduced to $|V\backslash J| = m$ dimensions. Using the Lovász extensions of the set functions that appear in the above objective function, we obtain the following tight relaxation:

$$\min_{f\in\mathbb{R}^m_+} \frac{R_1(f)-R_2(f)}{S_1(f)-S_2(f)}$$

where

$R_1(f) = \langle (g_i)_{i=1}^m + \gamma\left((h_i)_{i=1}^m\right), f\rangle + \mathrm{vol}_g(J) f_{\max}$, $R_2(f) = \gamma T_{k_1',k_2'}(f)$, $S_1(f) = \langle (d_i)_{i=1}^m + (d_i^{(J)})_{i=1}^m, f\rangle + \mathrm{assoc}(J) f_{\max}$ and $S_2(f) = \frac{1}{2}\sum_{i,j}^m w_{ij}\,|f_i - f_j|$ with $f_{\max} = \max_{i=1,\dots,m} f_i$.

RatioDCA

The original `Ratio-DCA`, originally proposed by [6], aims at minimizing a non-negative ratio of one-homogeneous difference of convex functions. In our case we apply the same algorithm, with a further modification that lies in the inner problem where we consider the constraint on the positive orthant. The scheme of `Ratio-DCA`[3] is defined as follows:

```
Initialization  $f^0 \in \mathbb{R}^n_+, \lambda^0 = Q\left(f^0\right)$
repeat
    $f^{l+1} = \underset{u\in\mathbb{R}^n_+,\|u\|_2\le 1}{\arg\min} \{R_1(u) - \langle u, r_2(f^l)\rangle +$
        $+ \lambda^l\left(S_2(u) - \langle u, s_1(f^l)\rangle\right)\}$
    where $r_2(f^l) \in \partial R_2(f^l), s_1(f^l) \in \partial S_1(f^l)$
    $\lambda^{l+1} = Q\left(f^{l+1}\right)$
until $\frac{|\lambda^{l+1}-\lambda^l|}{\lambda^l} < \epsilon$
```

In our problem this process is performed until all constraints are satisfied, that is by increasing the parameter γ sequentially whenever the solution obtained is not feasible.

Constrained local community detection

The performance of RatioDCA depends on how we solve the corresponding inner problem, which form, for the tight relaxation introduced previously, is defined as:

$$\min_{f\in\mathbb{R}^m_+,\|f\|_2\le 1} \left\{c_1 f_{\max} + \langle f, c_2\rangle + \lambda^l \frac{1}{2}\sum_{i,j}^m w_{ij}\,|f_i - f_j|\right\}$$

From [2], the exact formulation for $c_1 \in \mathrm{R}$ and $c_2 \in \mathrm{R}^m$ is known, since the same method and structure of the problem are considered. Specifically, their expressions can be written as:

$$\begin{aligned} c_1 &= \mathrm{vol}_g(J) + \gamma k_1' - \lambda^l\,\mathrm{assoc}(J) \\ c_2^l &= (g_i)_{i=1}^m + \gamma((h)_i)_{i=1}^m - \gamma(t^{(2)}_{k_1'}(f) + \\ &\quad + t^{(2)}_{k_2'}(f)) - \lambda^l((d_i)_{i=1}^m + (d_i^{(J)})_{i=1}^m) \end{aligned}$$

where for a specific k' the term $t^{(2)}_{k'}$ indicates an element of the subdifferential for $T^2_{k'}$ and $d_i^{(J)} = \sum_{j\in J} w_{ij}$.

The dual problem for the above inner problem is:

$$-\min_{\|\alpha\|_\infty\le 1}\min_{v\in S_m} \Psi(\alpha) := \frac{1}{2}\left\|P_{\mathrm{R}^m_+}\left(-c_1 v - c_2 - \frac{\lambda^l}{2}A\alpha\right)\right\|_2^2$$

where $(A\alpha)_i := \sum_j w_{ij}\,(\alpha_{ij} - \alpha_{ji})$, $P_{\mathrm{R}^m_+}$ represents the projection on the positive orthant and S_m is the simplex, defined as $S_m = \{v \in \mathbb{R}^m \mid v_i \ge 0, \sum_{i=1}^m v_i = 1\}$.

This dual problem can be solved with FISTA [1] (*Fast iterative shrinkage-thresholding algorithm*), a proximal gradient method which ensures a quadratic convergence rate $\mathcal{O}(\frac{1}{k^2})$ where k is the number of iterations. In the implementation this algorithm was tested using a fixed stepsize $\frac{1}{L}$ where $L > 0$ is the Lipshitz constant of $\nabla\Psi$, where an upper bound on L of $\nabla\Psi$ is given in [2] by $2\max_i \sum_{j|(i,j)\in E} w_{ij}^2$. As a starting point we need do consider $t_1 = 1$ and $\alpha_0, \beta_0 \in \mathbb{R}^{|E|}$. In order to solve the minimization problem, given in the first line in the scheme of FISTA[3]

$$v = \underset{u\in S_m}{\arg\min}\left\|P_{\mathbb{R}^m_+}\left(-c_1 u - c_2 - \frac{\lambda^l}{2}A\alpha\right)\right\|_2^2$$

it is useful to consider the following lemma, given in [3], which allows to reduce the problem with a standard projection onto the simplex:

$$\underset{v\in S_m}{\arg\min}\|y - v\|_2^2 \in \underset{v\in S_m}{\arg\min}\left\|P_{\mathbb{R}^m_+}(x - v)\right\|_2^2$$

with $x \in \mathbb{R}^m$ and $y := P_{\mathbb{R}^m_+}(x)$. In literature this is a well studied problem and in practice there are different algorithms one can consider. In general the projection itself simply aims to apply the following operation $x_i = \max\{y_i - \tau, 0\}$. So the crucial step is to find a value of τ such that $\sum_{i=1}^{m} \max\{y_i - \tau, 0\} = 1$. In the implementation it was considered an algorithm, explained in [4], based on sorting the elements of y in decreasing order into $u : u_1 \geq \cdots \geq u_m$ and identifying a set K such that:

$$K := \max_{1 \leq k \leq N} \{k | (\sum_{r=1}^{k} u_r - 1) \leq u_k\}$$

Set $\tau := \left(\sum_{k=1}^{K} u_k - 1\right) / K$ allows to perform the projection onto the simplex; the complexity of this algorithm is $\mathcal{O}(n \log n)$ in the worst case. After this step, as explained in [3], FISTA follows the following iterates:

$$z = P_{\mathbb{R}_+^m}\left(-c_1 v - c_2 - \frac{\lambda^l}{2} A\alpha\right)$$
$$\beta_{rs}^{k+1} = P_{B_\infty(1)}\left(\alpha_{rs}^k + \frac{1}{L}\lambda^l w_{rs}\left(z_r - z_s\right)\right)$$
$$t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}$$
$$\alpha_{rs}^{k+1} = \beta_{rs}^{k+1} + \frac{t_k - 1}{t_{k+1}}\left(\beta_{rs}^{k+1} - \beta_{rs}^k\right)$$

where $P_{B_\infty(1)}(x) = \min\{\max\{x, -1\}, 1\}$ and $B_\infty(1) = \{x \in \mathbb{R} : |x| \leq 1\}$. As stopping criterion, first we check if the difference between two consecutive β iterates is smaller respect to a small constant $\epsilon > 0$, otherwise continue until the maximum number of iterations (set to 2000) is reached.

Part IV
Experiments

The performances of `Ratio-DCA` and `Fast-atvo` are evaluated on 3 collaborative network datasets[8] of increasing size:

- `CA-GrQc`: $(|V|, |E|) = (5242, 14496)$
- `CA-HepTh`: $(|V|, |E|) = (9877, 25998)$
- `Cit-HepTh`: $(|V|, |E|) = (27770, 352807)$

In the following, the main results for `Ratio-DCA` and `Fast-atvo` are presented (full results for both methods can be found in *results.csv*). The results for both algorithms have been obtained using the leading eigenvector of the modularity matrix as starting point. `Fast-atvo` returns Q^*, the highest modularity, and S^*, the corresponding set. From S^* is possible to compute the set functions *NCut* and *density*, allowing a direct comparison with `Ratio-DCA`.
The key observations are:

1. the relation between k_2 (upper bound) and CPU runtime
2. the relation between $dist_{max}$ (max distance of nodes from set seed J) and CPU runtime
3. the relation between $dist_{max}$ and k_2
4. the relation between the choice and size of J and CPU runtime
5. the relation between k_2 and γ (the coefficient of constraint)
6. comparison between results from `Ratio-DCA` and `Fast-atvo`

Observations

In terms of CPU runtime, by looking at tables 1 and 2, it would seem that the value of k_2 tends to speed up the execution time to find the optimal solution, this happens when you fix an increasingly larger upper bound k_2 for the cardinality. So in such scenarios it seems more likely that there is a lower penalty for the introduced constraint, in terms of γ value, and that the constraint itself is respected.
On the other hand a larger distance $dist_{max}$, for a fixed value of k_2, leads to find the optimal solution at the cost of an increased CPU time, this may depend either on the seed set considered or probably on some local structure in the graph that affects the method. When the value of k_2 is small, the time required is pretty high and the γ tends to increase, therefore more values of γ are considered during the optimization phase, meaning that is more difficult to satisfy the constraint immediately.
For what concerns the selection of the seed sets, we selected sets whose nodes didn't have a too high degree, so with intermediate values, in order to avoid selecting nodes with many connections or that are isolated [9]. In this context the choice of the seed set is a further additional information we want to introduce in our problem, so by changing

[8]These datasets are the first 3 ones used in [3]

the initial J we expect different results. For example defining a seed with a hub (node with the greatest degree) we expect to find a result that would tend to include more communities, obtaining an unrepresentative cluster in the network. For every dataset, used in the implementation of `Ratio-DCA`, we defined different seed sets with different size. In addition, the minimum cluster size, that one wants to get in the final solution, is not known in advance. In general, this choice should be based on additional prior information regarding which value of lower bound k_1 to test. Since we don't have enough prior knowledge about that value we tried several upper bounds k_2 on the cardinality by keeping k_1 fixed, as specified in Tables 1 and 2.
Based on the results we have obtained, it seems that the method using `Ratio-DCA` is inefficient and very expensive, in terms of execution time, for large data sets (as `Cit-HepTh`) when we generally impose an upper bound constraint less than or equal to 20.
`Fast-atvo` provides a *global* but *loose* solution (since it's based on the smooth approximation TV_Q^p). `Ratio-DCA` provides a *local* but *tight* solution. Hence the first:

- is more robust (less prone to get stacked in local optima regardless of starting solution)
- does not require the use of any prior information
- discovers the leading community over the whole graph exploiting the discrepancies with respect to the expected edges' weights of a *null* model

The latter:

- is less robust (found solution heavily depends on the *set seed* J)
- requires the use of prior information (J, k_1, k_2, $dist_{max}$), that, if available, are useful to lead the algorithm faster toward the solution, but if not, increases the CPU runtime.
- discover the leading community *around* the set seed J

References

[1] Amir Beck and Marc Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM journal on imaging sciences* 2.1 (2009), pp. 183–202.

[2] Thomas Bühler. "A flexible framework for solving constrained ratio problems in machine learning". In: (2014).

[3] Thomas Bühler et al. "Constrained fractional set programs and their application in local clustering and community detection". In: *30th International Conference on Machine Learning, ICML 2013* (June 2013).

[4] Laurent Condat. "Fast projection onto the simplex and the $\boldsymbol{l_1}$ ball."". In: *Mathematical Programming* 158.1-2 (2016), pp. 575–585.

[5] P. Mercado F. Tudisco and M. Hein. "Community detection in network via nonlinear modularity eigenvectors". In: *SIAM Journal of Applied Mathematics* (2018).

[6] Matthias Hein and Simon Setzer. "Beyond spectral clustering-tight relaxations of balanced graph cuts". In: *Advances in neural information processing systems.* 2011, pp. 2366–2374.

[7] F. Lampariello L. Grippo and S. Lucidi. "A non monotone line search technique for newton's method". In: *SIAM Journal on Numerical Analysis* (1986).

[8] Michael Mahoney, Lorenzo Orecchia, and Nisheeth Vishnoi. "A Local Spectral Method for Graphs: With Applications to Improving Graph Partitions and Exploring Data Graphs Locally". In: *Journal of Machine Learning Research* 13 (2012).

[9] F. Moradi, T. Olovsson, and P. Tsigas. "A local seed selection algorithm for overlapping community detection". In: *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014).* 2014, pp. 1–8.

[10] M. E. J. Newman. "Finding community structure in networks using the eigenvectors of matrices". In: *Phys. Rev.* (2006).

RATIO DCA

DATASET	k_1	k_2	$dist_{max}$	seed	density	NCut	CPU(seconds)	γ
CA-GrQc	1	5	2	[2535]	0.0851	0.0074	51.2	10
CA-GrQc	1	10	2	[2535]	0.2022	0.0122	27.1	5
CA-GrQc	1	20	2	[2535]	0.4395	0.0167	10.9	2
CA-GrQc	1	60	2	[2535]	0.9711	0.0017	6.0	1
CA-GrQc	1	80	2	[2535]	0.9711	0.0017	3.8	1
CA-HepTh	1	5	2	[62227]	0.0336	0.0044	339.1	7
CA-HepTh	1	10	2	[62227]	0.0566	0.0083	186.3	4
CA-HepTh	1	20	2	[62227]	0.1324	0.0144	102.8	2
CA-HepTh	1	60	2	[62227]	0.2683	0.0296	98.4	1
CA-HepTh	1	80	2	[62227]	0.3055	0.0328	90.0	1
Cit-HepTh	1	5	2	[9907129]	-	-	>14000	-
Cit-HepTh	1	10	2	[9907129]	-	-	>14000	-
Cit-HepTh	1	20	2	[9907129]	0.013	0.021	12300	4
Cit-HepTh	1	60	2	[9907129]	0.0370	0.0357	21.2	2
Cit-HepTh	1	80	2	[9907129]	0.0477	0.0420	37.9	2

Table 1: single seed

RATIO DCA

DATASET	k_1	k_2	$dist_{max}$	seed	density	NCut	CPU(seconds)	γ
CA-GrQc	1	5	2	[19433, 5209]	0.0581	0.0112	416.2	12
CA-GrQc	1	10	2	[19433, 5209]	0.1304	0.0207	234.4	6
CA-GrQc	1	20	2	[19433, 5209]	0.2999	0.0306	150.2	3
CA-GrQc	1	60	2	[19433, 5209]	0.3617	0.0145	7.3	1
CA-GrQc	1	80	2	[19433, 5209]	0.3617	0.0145	5.1	1
CA-HepTh	1	5	2	[61742, 55387]	0.0596	0.0043	163.1	3
CA-HepTh	1	10	2	[61742, 55387]	0.1093	0.0075	113.6	2
CA-HepTh	1	20	2	[61742, 55387]	0.1745	0.0115	71.6	1
CA-HepTh	1	60	2	[61742, 55387]	0.2366	0.0306	71.0	1
CA-HepTh	1	80	2	[61742, 55387]	0.2622	0.0325	77.1	1
Cit-HepTh	1	5	2	[5905,5080]	-	-	>14000	-
Cit-HepTh	1	10	2	[5905,5080]	-	-	>14000	-
Cit-HepTh	1	20	2	[5905,5080]	-	-	>14000	-
Cit-HepTh	1	60	2	[5905,5080]	0.0708	0.0207	3180	2
Cit-HepTh	1	80	2	[5905,5080]	0.0708	0.0207	3192	2

Table 2: double seeds

FAST-ATVO

DATASET	density	Ncut	CPU(seconds)
CA-GrQc	0.9591	0,0000138	84
CA-HepTh	0.9054	0.0000351	320
Cit-HepTh	0.9557	0.000000029	1974

Table 3: fast-atvo