UNIVERSITÀ DEGLI STUDI DI PADOVA

Università degli Studi di Padova

Padua Research Archive - Institutional Repository

Total variation based community detection using a nonlinear optimization approach

(Article begins on next page)

Total variation based community detection using a nonlinear optimization approach

Andrea Cristofari*, Francesco Rinaldi*, Francesco Tudisco†

*Department of Mathematics "Tullio Levi-Civita"
University of Padua
Via Trieste 63, 35121 Padua (Italy)
E-mail: `andrea.cristofari@unipd.it`, `rinaldi@math.unipd.it`

†Gran Sasso Science Institute
Viale F. Crispi 7, 67100 L'Aquila (Italy)
E-mail: `francesco.tudisco@gssi.it`

Abstract. Maximizing the modularity of a network is a successful tool to identify an important community of nodes. However, this combinatorial optimization problem is known to be NP-complete. Inspired by recent nonlinear modularity eigenvector approaches, we introduce the modularity total variation TV_Q and show that its box-constrained global maximum coincides with the maximum of the original discrete modularity function. Thus we describe a new nonlinear optimization approach to solve the equivalent problem leading to a community detection strategy based on TV_Q. The proposed approach relies on the use of a fast first-order method that embeds a tailored active-set strategy. We report extensive numerical comparisons with standard matrix-based approaches and the Generalized RatioDCA approach for nonlinear modularity eigenvectors, showing that our new method compares favourably with state-of-the-art alternatives.

Keywords. Community detection. Graph modularity. Total variation. Nonlinear optimization. Active-set method.

MSC2000 subject classifications. 49M20. 65K10. 91D30. 91C20.

1 Introduction

Identifying important communities in a complex network is a difficult and highly relevant problem which has applications in different disciplines, including social network analysis [43, 49], molecular biology [35], politics [50], computer science [53] and many more (see e.g. [46]). Community detection can be used as a way to highlight so–called mesoscale properties of a network and further investigation into such communities may allow to gain insights about the network structure and the behavior of processes that take place on the network.

There is a large literature on the subject, with numerous different definitions of community and associated community detection methods [27, 39]. In this work a community is roughly understood as a set of nodes being highly connected inside and poorly connected with the rest of the graph. Although this high-level notion of community is fairly easy to understand and

broadly shared in the literature, a great variety of rigorous definitions have been proposed to make the community detection task mathematically and algorithmically precise, including notions involving edge-counting, random walk trapping, information theory and generative models such as stochastic block models, see e.g. [28]. A popular and successful idea, originally introduced by Newman and Girvan in [47], is based on the optimization of the modularity quality function.

The modularity function of a set of nodes S quantifies the difference between the actual and expected weight of edges in S, if edges were placed at random according to a random *null model*. The set S is then typically identified as a community if its modularity is "large enough". Thus, the definition of modularity function depends on the choice of the random model to which the actual network is compared to. Although various choices for such a null model have been considered in the literature [24, 55], the arguably most popular and successful choice is based on the configuration model or Chung-Lu random graph [14]. This is the null model we will always assume throughout this work.

Before providing the definition of modularity, let us introduce the notation and terminology that will be used throughout the whole paper.

Given a vector x, we denote by x_i the ith component of x. Given a matrix M, we denote by M_{ij} the entry of M having position (i, j). Similarly, given a vector x and a set of indices $I \subseteq \{1, \ldots, n\}$, we indicate by x_I the subvector of x with components x_i, $i \in I$. Further, the gradient of a function $f \colon \mathbb{R}^n \to \mathbb{R}$ at x is a column vector denoted by $\nabla f(x)$.

Now, let $G = (V, E)$ be an undirected network with node set $V = \{1, \ldots, n\}$ and nonnegative weight matrix $A = (A_{ij} \geq 0)$ such that $A_{ij} > 0$ if and only if $ij \in E$. Let d be the vector of (weighted) degrees, $d_i = \sum_j A_{ij}$, and let $\operatorname{vol} G = \sum_{i=1}^{n} d_i$ be the graph volume.

Definition 1 (Modularity of a set). *Let $S \subseteq V$ be any set of nodes. The quantity*

$$Q(S) = \frac{1}{\operatorname{vol} G} \sum_{i,j \in S} \left(A_{ij} - \frac{d_i d_j}{\operatorname{vol} G} \right)$$

is the modularity measure of the set S.

The rank-one term $d_i d_j / \operatorname{vol} G$ is the one responsible for the null model and it has to be interpreted as the expected weight of edges in the graph, if edges were placed independently at random according to the Chung-Lu model. The term A_{ij}, instead, is the actual weight of the edges and so, when $Q(S) > 0$, the weight of edges in S exceeds the expected weight. We call such an S a module. Modules with large value of modularity are typically good indicators of communities in a network. On the other hand, not all the subsets with positive modularity form a community and the range of attainable values of the modularity function may vary significantly from network to network, see e.g. [15, 33]. While discerning whether a module is or is not a community is itself an interesting research question, this issue goes beyond the scope of this work. Here we assume that the "most important" module in G is a decent indicator of a community and we formally state the following leading module problem:

Definition 2 (Leading module problem). *Find $S^* \subseteq V$ such that*

$$Q(S^*) = \max_{S \subseteq V} Q(S) =: Q^* \,. \tag{1}$$

Note that, with Definition 1, if S is a module, then also its complement $\bar{S} = V \setminus S$ is so. In fact, a simple computation reveals that

$$Q(\bar{S}) = Q(S)$$

for all $S \subseteq V$. In particular, both $Q(V)$ and $Q(\emptyset)$ are zero. So the leading module problem can be equivalently thought of as a leading bipartition problem, where one assesses the bipartition as a whole rather than the individual community. Already this simple observation shows that the solution to (1) is not unique. In general it is well known that very dissimilar sets S may yield optimal (or nearly optimal) modularity values $Q(S)$ [29], and we are interested in computing any of them.

From Definition 2, the leading module problem boils down to a combinatorial optimization problem which is known to be strongly NP-complete [9]. Different strategies have been proposed to compute an approximate solution. Linear relaxation approaches are based on the spectrum of specific matrices (as the modularity matrix or the Laplacian matrix) and have been widely explored and applied to various research areas, see e.g. [23, 44, 45, 54]. Computational heuristics have been developed for optimizing directly the discrete quality function (see e.g. [39, 50]), including for example greedy algorithms [16], simulated annealing [33] and extremal optimization [21]. Among them, the locally greedy algorithm known as Louvain method [7] is arguably the most popular one.

However, most of the times, the proposed approaches are meant to address the multi-community problem where, rather than just Q, one aims at maximizing the sum $\sum_k Q(S_k)$ over the set of all possible partitions $\{S_1, S_2, \ldots, \}$ of V. As the number of sets forming the partition is not initially prescribed, this problem is intrinsically different from the leading module problem and adapting the various approaches is most of the times either not trivial or not possible. A model that is gaining increasing popularity in recent years and that adapts very well to the leading module case is based on nonlinear relaxation. This approach exploits the connection between modularity optimization and nonlinear eigenvalue problems [11, 36, 56, 57], on the one hand, and with total variation and image processing [8, 10, 38], on the other. In particular, it is proven in [58] that the optimization problem (1) can be equivalently recast into the optimization of a function of the type

$$r(x) = \frac{f(x)}{g(x)} \tag{2}$$

where f and g are suitable continuous functions of real variables. As the two problems of maximizing Q and maximizing r coincide, this process is called *nonlinear exact relaxation* of Q [58]. The advantage of this approach is that, when a local maximum of r is computed, the corresponding maximizer can be used to identify a community of nodes in the network that consistently outperforms standard approaches based on non-exact relaxations.

Furthermore, this result shows that one can recast the modularity combinatorial optimization problem (1) as a nonlinear eigenvector problem and approach it using strategies for the solution of nonlinear eigenproblems. In particular, inspired by the inverse power method for matrices, an algorithm called Generalized RatioDCA is proposed in [58] for optimizing the ratio of functions in (2). Such method is an extension of the RatioDCA method originally developed in the context of unsupervised learning [37]. Each iteration of Generalized RatioDCA consists of two substeps that roughly go as follows: the first step (outer iteration) is based on the observation that any

3

critical point x of r is such that $0 \in \partial f(x) - \lambda \partial g(x)$, for some coefficient $\lambda \in \mathbb{R}$ and where ∂ denotes the subgradient. This, combined with the properties of the subgradient, allows us to transform the optimization of $r(x)$ into the optimization of a new function $\tilde{r}(x)$ being defined as the difference of convex functions; in the second step (inner iteration) the new function $\tilde{r}$ is optimized via a primal-dual algorithm such as FISTA [5] or PDHG [12]. A convergence result is proposed in [58], showing that the generated sequence eventually converges to a stationary point of r. The main drawback of this approach is that the inner-outer optimization method proposed can be computationally demanding and thus makes this approach somewhat prohibitive for large datasets.

The contribution of this work is twofold:

1. We show, in Section 2, that the leading community problem (1) is equivalent to the box-constrained optimization of the so–called "modularity total variation", a particular graph total variation function with positive and negative weights. This result, presented in Theorem 1, improves the one of [58] and shows that (1) is equivalent to the problem of maximizing a continuous real-valued function $f(x)$ subject to an arbitrary box constraint of the type $-a \le x_i \le b$, $a, b > 0$. Moreover, it provides a simpler and self-contained proof of such equivalence result. In other words, this result unveils the intriguing equivalence between the unconstrained combinatorial leading module optimization problem and the constrained but continuous optimization of $f(x)$ and thus it allows us to efficiently attack the leading module problem with techniques from continuous optimization.

2. Based on this theorem, in Section 3 we propose an approach to optimize the modularity total variation function f. Precisely, we first modify the nonsmooth total variation by considering an approximation $\tilde{f}$ with continuous derivatives, and then we maximize it using an algorithmic framework that embeds an active-set first-order method for box-constrained problems, named Fast Active-SeT based Approximate Total Variation Optimization algorithm (`FAST-ATVO`). We prove global convergence of `FAST-ATVO` and provide extensive experiments which show that the computational performance and modularity quality score obtained with the proposed approach compare favorably with the classical linear spectral method [45] and the Generalized RatioDCA method described in [58]. The `FAST-ATVO` software that we developed is publicly available, as pointed out in Section 4.

The paper is structured as follows: In the next section we introduce the concept of modularity total variation, we review the linear and nonlinear spectral methods for modularity maximization and we prove our first main result, showing the equivalence between (1) and a continuous box-constrained optimization problem. Then, in Section 3, we give a detailed description of the optimization approach, introduce the general scheme of `FAST-ATVO` and analyze its global convergence. In Section 4, we report extensive numerical experiments on real–world data to show the performance of our new method and discuss the complexity of the algorithm in more detail. Finally, we draw some conclusions in Section 5.

2 Exact modularity relaxation via total variation

The total variation is classically defined for continuous functions: Given a real smooth function $u : X \to \mathbb{R}$ with $X \subseteq \mathbb{R}^n$, one defines $TV_X(u) = \int_X |\nabla u|$. The extension to the graph setting $G = (V, E)$ and to graph valued functions $u : V \to \mathbb{R}$ (which we always implicitly identify

with vectors in $\mathbb{R}^n$) is done by replacing the gradient with the nonlocal directional derivative $\nabla u : E \to \mathbb{R}$, which measures the variation of u along the edge $ij \in E$ via $\nabla u(ij) = A_{ij}(u_i - u_j)$. The graph total variation of u is then

$$TV_G(u) = \frac{1}{2} \sum_{ij} |\nabla u(ij)| = \frac{1}{2} \sum_{ij} A_{ij} |u_i - u_j| \tag{3}$$

where, since the graph is undirected, the $1/2$ is required due to the fact that the nonzero entries of the adjacency matrix A account for twice the number of edges.

In the graph setting, the total variation (3) is strictly related with the concept of Lovász extension of a set valued function $F : 2^V \to \mathbb{R}$, which we recall in the following

Definition 3. *Consider a function $F : 2^V \to \mathbb{R}$. Given a vector $x \in \mathbb{R}^n$, reorder it so that $x_1 \leq x_2 \leq \ldots \leq x_n$ and let $C_i \subseteq V$ be the set $C_i = \{k \in V : x_k \geq x_i\}$. The Lovász extension $f_F : \mathbb{R}^n \to \mathbb{R}$ of F is defined by*

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

The relation between total variation and Lovász extension is through the set-valued cut function $K : 2^V \to \mathbb{R}$ defined by $K(S) = \sum_{i \in S, j \notin S} A_{ij}$. In fact, we have

Lemma 1. *For any graph G and any $x \in \mathbb{R}^n$ it holds $f_K(x) = TV_G(x)$.*

Proof. Assume for simplicity, and without loss of generality, that x is such that $x_1 \leq \cdots \leq x_n$. Since by definition $K(V) = K(\emptyset) = 0$, we have

$$\begin{aligned}
f_K(x) &= \sum_{i=1}^{n-1} K(C_{i+1})(x_{i+1} - x_i) + K(V)x_1 \\
&= \sum_{i=1}^{n-1} |x_{i+1} - x_i| \sum_{k \in C_{i+1}, h \notin C_{i+1}} A_{hk} \\
&= \sum_{i=1}^{n-1} |x_{i+1} - x_i| \sum_{k \geq s(i+1), h < s(i+1)} A_{hk} \\
&= \sum_{k \geq h} A_{kh} |x_k - x_h| = TV_G(x)
\end{aligned}$$

where, for $i \in V$, $s(i) \in V$ denotes the smallest index such that $x_k \geq x_i$ for all $k \geq s(i)$. $\qquad\square$

The following result thus relates the total variation with the Lovász extension of the modularity function.

Lemma 2. *Let G and G_0 be the graphs with adjacency matrices $A = (A_{ij})$ and $B = (d_i d_j / \operatorname{vol} G)$, respectively. Then*

$$f_Q(x) = \frac{TV_{G_0}(x) - TV_G(x)}{\operatorname{vol} G} = \frac{1}{2 \operatorname{vol} G} \sum_{ij} \left(\frac{d_i d_j}{\operatorname{vol} G} - A_{ij} \right) |x_i - x_j|$$

for all $x \in \mathbb{R}^n$.

5

Proof. From the definition of $Q(S)$ we have

$$Q(S) = \frac{1}{\text{vol}\,G}\Big(\sum_{ij \in S} A_{ij} - \frac{d_i d_j}{\text{vol}\,G}\Big) = \frac{1}{\text{vol}\,G}\Big(\sum_{ij \in S} d_i - \frac{d_i d_j}{\text{vol}\,G} - (d_i - A_{ij})\Big). \tag{4}$$

Using the identities $d_i = \sum_{k \in V} A_{ik} = \sum_{k \in V} d_i d_k / \text{vol}\,G$, we get

$$\sum_{ij \in S} d_i - \frac{d_i d_j}{\text{vol}\,G} = \sum_{ij \in S}\sum_{k \in V} \frac{d_i d_k}{\text{vol}\,G} - \frac{d_i d_j}{\text{vol}\,G}$$

$$= \sum_{i \in S}\Big(\sum_{k \in V} \frac{d_i d_k}{\text{vol}\,G} - \sum_{j \in S} \frac{d_i d_j}{\text{vol}\,G}\Big) = \sum_{i \in S, j \notin S} \frac{d_i d_j}{\text{vol}\,G}$$

and similarly

$$\sum_{ij \in S} d_i - A_{ij} = \sum_{ij \in S}\sum_{k \in V}(A_{ik} - A_{ij}) = \sum_{i \in S, j \notin S} A_{ij}.$$

Thus, plugging the last two identities in (4) we get

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

where $K_0(S) = \sum_{i \in S, j \notin S} B_{ij}$ is the cut function of the graph G_0 with adjacency matrix $B_{ij} = d_i d_j / \text{vol}\,G$. From Lemma 1 and the linearity of the Lovász extension, $f_{F_1 + F_2} = f_{F_1} + f_{F_2}$, we conclude. $\square$

So the Lovász extension of Q is the difference of two total variations or, equivalently, it is the total variation on the graph $G^{\pm}$ with real valued adjacency matrix $M = B - A = (d_i d_j / \text{vol}\,G - A_{ij})$, 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 $M_{ij} \neq 0$. As this function is the main tool of our analysis we call it *modularity total variation* and denote it from now on as

$$TV_Q(x) = \frac{1}{2}\sum_{ij}\Big(\frac{d_i d_j}{\text{vol}\,G} - A_{ij}\Big)|x_i - x_j|.$$

Let $\mathbb{1}_S$ be the binary vector $(\mathbb{1}_S)_i = 1$ if $i \in S$ and $(\mathbb{1}_S)_i = 0$ otherwise. It is well known that, for any set valued function $F : 2^V \to \mathbb{R}$, it holds $f_F(\mathbb{1}_S) = F(S)$ (see e.g. [3]). Therefore, from Lemma 2, $TV_Q(\mathbb{1}_S) / \text{vol}\,G = f_Q(\mathbb{1}_S) = Q(S)$. However, for the specific modularity function, the following additional formula holds

Lemma 3. *Let $S \subseteq V$ and let $a, b \in \mathbb{R}$. We have*

$$TV_Q(a\mathbb{1}_S + b\mathbb{1}_{\bar{S}}) = \text{vol}\,G \cdot |a - b| \cdot Q(S).$$

Proof. Let $M_{ij} = B_{ij} - A_{ij} = d_i d_j / \text{vol}\,G - A_{ij}$. As $|x_i - x_j| = 0$ when $x = a\mathbb{1}_S + b\mathbb{1}_{\bar{S}}$ and i, j are either both in S or both in $\bar{S}$, we have

$$2 \cdot TV_Q(a\mathbb{1}_S + b\mathbb{1}_{\bar{S}}) = \sum_{i \in S, j \notin S} M_{ij}|a - b| + \sum_{i \notin S, j \in S} M_{ij}|b - a|. \tag{5}$$

6

Now notice that $\sum_{i \in S, j \notin S} M_{ij} = \sum_{i \in S, j \in V} M_{ij} - \sum_{i,j \in S} M_{ij}$, moreover

$$\sum_{j \in V} M_{ij} = \frac{d_i}{\mathrm{vol}\, G} \sum_{j \in V} d_j - \sum_{j \in V} A_{ij} = d_i - d_i = 0\,.$$

Combined with (5) we get

$$\frac{TV_Q(a\mathbb{1}_S + b\mathbb{1}_{\bar{S}})}{\mathrm{vol}\, G\, |a - b|} = \frac{1}{\mathrm{vol}\, G} \sum_{ij \in S} \left(A_{ij} - \frac{d_i d_j}{\mathrm{vol}\, G} \right) = Q(S)\,,$$

concluding the proof. $\qquad\qquad\square$

Lemma 3 shows that the modularity total variation TV_Q somewhat interpolates the modularity set valued function Q on vectors having only two different values across all the entries. Note that all these vectors identify uniquely a set of nodes and, in fact, there is a one-to-one correspondence between vectors of the type $a\mathbb{1}_S + b\mathbb{1}_{\bar{S}}$ and the subsets of V. However, a deeper and more relevant property of the modularity total variation TV_Q is that its maximum coincides (up to scaling) with the maximum of Q. This was proved in [58] and is the reason why TV_Q is therein called a *nonlinear exact relaxation* of Q. We generalize that result in the following theorem, where we also provide a simpler and self–contained proof.

Theorem 1. *Let $a, b > 0$ and let $x^* \in \mathbb{R}^n$ be a solution of*

$$\begin{aligned} \max \quad & TV_Q(x) \\ s.t. \quad & -a \le x_i \le b, \quad i = 1, \ldots, n. \end{aligned} \qquad (6)$$

Then $TV_Q(x^*) = \mathrm{vol}\, G \cdot (a + b) \cdot \max_{S \subseteq V} Q(S)$.

Proof. For ease of notation, and without loss of generality, we prove the theorem assuming $\mathrm{vol}\, G = 1$. Let $\mathcal{B}(a, b) = \{x \in \mathbb{R}^n : -a \le x_i \le b\}$. As $u_S = b\mathbb{1}_S - a\mathbb{1}_{\bar{S}} \in \mathcal{B}(a, b)$, from Lemma 3 we get

$$(a + b) \max_{S \subseteq V} Q(S) = \max_{S \subseteq V} TV_Q(u_S) \le \max_{u \in \mathcal{B}(a,b)} TV_Q(u)\,.$$

Vice-versa, recall the identities $TV_Q(x) = f_Q(x)$ and $Q(V) = 0$. For any $u \in \mathcal{B}(a, b)$ assume w.l.o.g. $u_1 \le \ldots \le u_n$. We have

$$\begin{aligned} TV_Q(u) &= \sum_{i=1}^{n-1} Q(C_{i+1})(u_{i+1} - u_i) \le \left(\max_{i=1,\ldots,n-1} Q(C_{i+1}) \right)(u_n - u_1) \\ &\le \left(\max_{i=1,\ldots,n-1} Q(C_{i+1}) \right)(a + b) \end{aligned}$$

where $C_i = C_i(u) = \{k : u_k \ge u_i\}$. Therefore

$$\max_{u \in \mathcal{B}(a,b)} TV_Q(u) \le (a + b) \max_{u \in \mathcal{B}(a,b)} \max_{i=1,\ldots,n-1} Q(C_{i+1}) = (a + b) \max_{S \subseteq V} Q(S)$$

which shows the reverse inequality. Thus the maximum of (6) coincides with $(a + b)$ times the maximum of Q, concluding the proof. $\qquad\square$

7

By Theorem 1 we can now transform the leading module problem (1) into the box constrained optimization problem (6), where we seek for the maximum of TV_Q over the box $\mathcal{B}(a,b) = \{x \in \mathbb{R}^n : -a \leq x_i \leq b, \ i = 1, \ldots, n\}$. Note that we can freely choose the parameters $a, b > 0$. Note moreover that the maximum x^* is always attained on the border of $\mathcal{B}(a,b)$, i.e. it is of the form $x^* = b\mathbb{1}_S - a\mathbb{1}_{\bar{S}}$. This is shown by the following

Corollary 1. *Let x^* be as in Theorem 1. Then $x^* \in \partial\mathcal{B}(a,b)$, the border of the box $\mathcal{B}(a,b)$.*

Proof. Suppose by contradiction that $x^* \notin \partial\mathcal{B}(a,b)$. Then, there exist $a', b' > 0$ with $-a < -a' < b' < b$ such that $x^* \in \mathcal{B}(a',b')$. From Theorem 1 we have

$$(a+b)\max_{S \subseteq V} Q(S) = TV_Q(x^*) \leq \max_{x \in \mathcal{B}(a',b')} TV_Q(x) = (a'+b')\max_{S \subseteq V} Q(S)$$

and this is not possible as $(a+b) > (a'+b')$. □

2.1 Comparison with the nonlinear spectral approach

Nonlinear modularity eigenvectors are defined in [58] as critical points of the Rayleigh quotient

$$r_Q(x) = \frac{TV_Q(x)}{\|x\|_\infty}$$

and nonlinear eigenvectors x^* corresponding to the largest eigenvalue $\lambda^* = r_Q(x^*)$ are there used to locate leading communities. Due to the homogeneity of TV_Q, it is simple to observe that, when $a = b = 1$, any maximizer of r_Q is a solution to (6) and vice-versa. In fact, note that for $a = b = 1$, we have $B(a,b) = \{x : \|x\|_\infty \leq 1\}$. Thus, as $TV_Q(\lambda x) = \lambda\, TV_Q(x)$, for any scalar $\lambda > 0$, we obtain

$$\max_{x \in \mathcal{B}(a,b)} TV_Q(x) \geq \max_{\|x\|_\infty = 1} TV_Q(x) = \max_{x \in \mathbb{R}^n} TV_Q\left(\frac{x}{\|x\|_\infty}\right) = \max_{x \in \mathbb{R}^n} r_Q(x)$$
$$\geq \max_{x \in \mathcal{B}(a,b)} r_Q(x) \geq \max_{x \in \mathcal{B}(a,b)} TV_Q(x)$$

which implies that all the above inequalities are indeed identities.

3 Description of the nonlinear optimization approach

In order to compute an approximate solution to (6), we first consider a smooth approximation of problem TV_Q. In particular, we replace the modularity total variation with the following smooth function:

$$TV_Q^p(x) = \frac{1}{2}\sum_{ij}\left(\frac{d_i d_j}{\mathrm{vol}\,G} - A_{ij}\right)|x_i - x_j|^p,$$

with $p > 1$. Hence, the new problem we want to solve has the form:

$$\begin{aligned} \max \quad & TV_Q^p(x) \\ \text{s.t.} \quad & -a \leq x_i \leq b, \quad i = 1, \ldots, n, \end{aligned} \tag{7}$$

8

with $a, b > 0$. Even if (7) is still a hard non-convex problem, we can now use first-order methods to compute a good approximate solution of the original problem (6). In particular, in view of Corollary 1, we expect that also for the smooth problem (7) good solutions lie on the boundary of the feasible set $\mathcal{B}(a, b)$. For this reason, we develop an active-set method that dynamically selects, at each iteration, a set of variables x_i to be safely fixed at the boundary, and then optimizes over the free subspace (i.e., the subspace of variables not fixed). Further, we propose a global optimization framework, which embeds the active-set algorithm, to improve the overall quality function score.

As we will see, even though TV_Q^p and TV_Q may differ when $p > 1$, using such a smooth approximation and the proposed optimization algorithm allows us to both

- significantly increase the value of TV_Q (or, equivalently, of Q), and

- significantly reduce the required computational time

with respect to the previously proposed Generalized RatioDCA.

In the following, we first describe our active-set algorithm (in Subsection 3.1) and then we report the global optimization framework (in Subsection 3.2). Finally, in Subsection 3.3 we carry out a complexity analysis of function and gradient computation, which is useful to design an efficient implementation of the algorithm.

3.1 An algorithm for the approximate total variation optimization

In order to describe the algorithm we use for solving problem (7), let us consider a general bound-constrained optimization problem of the following form:

$$\min \{f(x) : l \leq x \leq u\}, \tag{8}$$

where the inequalities between vectors are meant to hold componentwise, $f \in C^1(\mathbb{R}^n)$, $l, u \in \mathbb{R}^n$ and $l < u$. Clearly, problem (7) is a particular case of (8), obtained by setting $f(x) = -TV_Q^p(x)$ and $l_i = -a$, $u_i = b$, $i = 1, \ldots, n$.

From now on, given a vector $x \in \mathbb{R}^n$, let us indicate by $[x]^\sharp$ the projection of x onto the feasible set of problem (8). Since we consider a non-convex objective function f, we also need to recall the definition of stationarity: a point $x^* \in [l, u]$ is said to be *stationary* for problem (8) if

$$\|x^* - [x^* - \nabla f(x^*)]^\sharp\| = 0,$$

or equivalently,

$$\nabla_i f(x^*) = 0, \quad i \colon l_i < x_i^* < u_i, \tag{9a}$$
$$\nabla_i f(x^*) \geq 0, \quad i \colon x_i^* = l_i, \tag{9b}$$
$$\nabla_i f(x^*) \leq 0, \quad i \colon x_i^* = u_i. \tag{9c}$$

The algorithm we propose here belongs to the class of *active-set methods* (see, e.g., [17, 18, 19, 25, 34, 48] and references therein), where an estimate of the active (or binding) inequality constraints at the final solution is iteratively updated at each iteration. In box-constrained problems, the estimate gets in practice a subset of variables that can be fixed to the bounds at

each iteration. This reduces the complexity of the search direction step and it turns out to be extremely useful in our context, where good points are likely to lie on the boundary of the feasible set.

Using the stationarity conditions (9), we can define the following active and non-active set estimates:

$$A_l(x) = \{i : x_i = l_i, \nabla_i f(x) > 0\}, \tag{10a}$$
$$A_u(x) = \{i : x_i = u_i, \nabla_i f(x) < 0\}, \tag{10b}$$
$$N(x) = \{i : i \notin A_l(x) \cup A_u(x)\}. \tag{10c}$$

In particular, for every feasible point x, the sets $A_l(x)$ and $A_u(x)$ contain the indices of those variables we estimate to be active at the lower and the upper bounds, respectively, in the final solution returned by the algorithm. Vice–versa, $N(x)$ contains the indices of those variables we estimate to be non-active at the final solution. Note that, by a slight abuse of terminology, we say that a variable is *active* if one of its bound constraints is active.

By adapting classic results for active-set strategies (see, e.g., [20, 22]), it is easy to prove the following proposition, whose proof is omitted here for the sake of brevity.

Proposition 1. *Let x^* be a stationary point for problem* (8). *Then, there exists a neighborhood* $\Omega(x^*, \rho) = \{x : \|x - x^*\| \leq \rho\}$ *such that*

$$\{i : x_i^* = l_i, \nabla_i f(x^*) > 0\} \subseteq A_l(x) \subseteq \{i : x_i^* = l_i\},$$
$$\{i : x_i^* = u_i, \nabla_i f(x^*) < 0\} \subseteq A_u(x) \subseteq \{i : x_i^* = u_i\},$$

for each $x \in \Omega(x^*, \rho)$.

The detailed description of the proposed algorithm, that we name Fast Active-SeT based Approximate Total Variation Optimization (`FAST-ATVO`), can be found in Appendix A, together with its convergence analysis. In what follows, instead, we detail the main ideas behind `FAST-ATVO` and a sketch of its structure. First, let us emphasize the two features that, together with the active-set estimate given in (10), most characterize our algorithm:

(i) the use of a non-monotone stabilization technique [31] that allows us to avoid the objective function computation at every iteration;

(ii) the possibility to update only a subset of variables at every iteration.

Both the above algorithmic features are particularly well suited for our specific problem, since, as we will show in Subsection 3.3, the computation of TV_Q^p

- is in general computationally expensive (and then, it is convenient to avoid computing it at every iteration);

- can be implemented in an efficient way when a small subset of variables is changed from one iteration to another (and then, decomposition approaches are convenient, especially when the problem dimension is large).

10

In particular, note that computing TV_Q^p on a vector $x \in \mathbb{R}^n$ can be significantly more expensive than computing the modularity Q of a set $S \subseteq V$, especially for small sets S, as the two would be equivalent only if the number of nonzero entries of x is exactly $|S|$, which we cannot expect in general.

The main steps of FAST-ATVO are overviewed in Algorithm 1 (see Algorithm 4 in Appendix A for a detailed description).

Algorithm 1 FAST-ATVO(x^0) – short scheme

Given a feasible point x^0, set $k = 0$

While x^k is non-stationary for problem (8)

    Compute $A_l^k = A_l(x^k)$, $A_u^k = A_u(x^k)$ and $N^k = N(x^k)$

    Choose $W^k \subseteq N^k$, set $d_{A_l^k}^k = 0$, $d_{A_u^k}^k = 0$, $d_{N^k \setminus W^k}^k = 0$ and compute $d_{W^k}^k$

    Compute a stepsize α^k by a non-monotone stabilization strategy, set
$x^{k+1} = [x^k + \alpha^k d^k]^\sharp$ and $k = k + 1$

End while

We see that, at the beginning of every iteration k, we compute the active and non-active set estimates as in (10) and define

$$A_l^k = A_l(x^k), \quad A_u^k = A_u(x^k) \quad \text{and} \quad N^k = N(x^k).$$

Then, a (non-empty) working set $W^k \subseteq N^k$ is chosen according to a Gauss-Southwell-type (or greedy) rule and a first-order search direction d^k is computed such that $d_{A_l^k}^k = 0$, $d_{A_u^k}^k = 0$ and $d_{N^k \setminus W^k}^k = 0$. Namely, d^k is computed in the reduced variable subspace defined by W^k. Finally, a non-monotone stabilization strategy is used to compute a stepsize α^k and generate the new iterate $x^{k+1} = [x^k + \alpha^k d^k]^\sharp$.

The following theorem shows the global convergence of FAST-ATVO to stationary points.

Theorem 2. *Let $\{x^k\}$ be the sequence of points generated by FAST-ATVO. Then, either an integer $\bar{k} \geq 0$ exists such that $x^{\bar{k}}$ is a stationary point for problem (8), or else the sequence $\{x^k\}$ is infinite and every limit point x^* of the sequence is a stationary point for problem (8).*

Proof. See Appendix A. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\square$

3.2   A global optimization strategy for improving the modularity

Even though FAST-ATVO is able to exploit the structure of the modularity total variation optimization problem (7), we can only guarantee convergence to a stationary point. This is not surprising as optimizing the network modularity Q is in general strongly NP-complete [9] and thus, due to Theorem 1, we cannot expect convergence to global minimizers of (7). Although, in practice, stationary points obtained with FAST-ATVO often provide quality results, we can employ a global optimization strategy in order to further improve the quality of the final solution. Of course, this has an additional computational cost.

The arguably most elementary global optimization strategy one can use is *Multistart*, which consists of repeatedly running a local optimization algorithm from some randomly chosen starting

points and finally picking the best one. This is the strategy that is used in [58]. In our case, a few experiments showed that Multistart is quite inefficient. Therefore, in order to get a better graph partition without significantly increasing the CPU time, we propose here a slightly more sophisticated technique. In particular, we adopted a form of iterated local search/basin hopping strategy (see, e.g., [32, 40]) that we call Partition & Swap (`PS`). The scheme of `PS` is reported in Algorithm 2.

Algorithm 2 `PS`(x^0)

---

0 Given a feasible point x^0
1 Set $\bar{x}^0 = $ `FAST-ATVO`(x^0) and $k = 1$
2 For $k = 1, 2, \ldots$
3   Set $x^k = $ `SWAP`$(\bar{x}^{k-1})$
4   Set $y^k = $ `FAST-ATVO`(x^k)
5   If $TV_Q(y^k) < TV_Q(\bar{x}^{k-1})$
6   Set $\bar{x}^k = y^k$
7   Else
8   Set $\bar{x}^k = \bar{x}^{k-1}$
9   End if
10 End for

We see that `PS` consists in applying, at each iteration, a perturbation to the current stationary point (according to the `SWAP` strategy described later) and starting `FAST-ATVO` from the new perturbed point. When `FAST-ATVO` gets a better solution, in terms of modularity total variation, then $\bar{x}_k$ (the current best solution) is updated; otherwise, $\bar{x}^k$ is left unchanged and the procedure is repeated until a maximum number of iterations is reached.

For what concerns the `SWAP` strategy, it is based on a simple idea of shifting a given percentage of variables in $\bar{x}^k$ to the bound with the opposite sign. We illustrate the `SWAP` scheme in Algorithm 3.

Algorithm 3 `SWAP`(x)

---

0 Given a feasible point x, choose $\sigma \in [0, 100]$
1 Partition $\{1, \ldots, n\}$ in I_l and I_u, such that $x_i \leq 0$ if $i \in I_l$ and $x_i \geq 0$ if $i \in I_u$
2 Set $y = x$
3 Randomly pick $\sigma\%$ of indices from I_l and set the corresponding components y_i to u_i
4 Randomly pick $\sigma\%$ of indices from I_u and set the corresponding components y_i to l_i
5 Return y

3.3   Complexity analysis of function and gradient computation

When solving problem (8), a severe bottleneck for computational efficiency is represented by the time spent for computing the objective function and its gradient, since, as to be shown below, both of them have a cost that may grow quadratically with the problem dimension. Therefore, a complexity analysis of these computations needs to be carry out in order to understand the most efficient way to perform the above operations in practice.

Let $f(x)$ be the objective function of problem (8). Denoting by M the symmetric matrix such that $M_{ij} = d_i d_j / \operatorname{vol} G - A_{ij}$ for all $i, j = 1, \ldots$, we can express the objective function $f(x) = TV_Q^p(x)$ as

$$f(x) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} M_{ij} |x_i - x_j|^p \tag{11}$$

and its gradient as

$$\nabla_i f(x) = p \sum_{\substack{j=1 \\ j \neq i}}^{n} M_{ij} \operatorname{sign}(x_i - x_j) |x_i - x_j|^{p-1}, \quad i = 1, \ldots, n. \tag{12}$$

We see that every function evaluation and every gradient evaluation have a cost (in terms of number of arithmetic operations)

$$\mathcal{O}\left(\frac{n(n-1)}{2}\right), \tag{13}$$

respectively. In our implementation of FAST-ATVO, we use some tricks to reduce these costs by exploiting the structure of f and ∇f. For convenience of exposition, we first focus on the gradient computation.

Given any iterate x^k, assume that the working set W^k has been chosen and the successive iterate x^{k+1} has been produced (as $x^{k+1} = x^k + \alpha^k d^k$, with $\alpha^k > 0$ and $d_i^k = 0$ for all $i \notin W^k$). For any index $i = 1, \ldots, n$, let us define the two functions

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

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

We see that $\phi_i(x)$ and $\varrho_i(x)$ differ only in that the summation is over $j \in W^k$ in the first function and over $j \notin W^k$ in the second function. Using (12), clearly we have $\nabla_i f(x) = \phi_i(x) + \varrho_i(x)$ for all $i = 1, \ldots, n$. Now, consider any index $h \notin W^k$. Since $x_h^{k+1} = x_h^k$, we can write

$$\nabla_h f(x^{k+1}) = \phi_h(x^{k+1}) + \varrho_h(x^k) = \phi_h(x^{k+1}) + \nabla_h f(x^k) - \phi_h(x^k).$$

In other words, we can obtain $\nabla_h f(x^{k+1})$ by computing $\phi_h(x^{k+1})$ and $\phi_h(x^k)$ (since $\nabla_h f(x^k)$ is known at the currrent iteration). So, we can get $\nabla f(x^{k+1})$ as follows:

$$\nabla_i f(x^{k+1}) = \begin{cases} \phi_i(x^{k+1}) + \varrho_i(x^{k+1}), & i \in W^k \\ \phi_i(x^{k+1}) + \nabla_i f(x^k) - \phi_i(x^k), & i \notin W^k. \end{cases} \tag{14}$$

Now, we are interested in the cost of computing ∇f by (14) in order to compare this cost with the one needed by (12). First, we see that (14) requires to compute $\phi_i(x^{k+1})$ for all $i \in W^k$. The cost of this operation is the order of the number of pairs of indices belonging to W^k, that is

$$\mathcal{O}\left(\frac{|W^k|(|W^k| - 1)}{2}\right).$$

Then, we have the cost of computing $\varrho_i(x^{k+1})$ for all $i \in W^k$ and $\phi_i(x^{k+1})$ for all $i \notin W^k$, which is the order of the number of pairs of indices such that exactly one of them belongs to W^k. The number of these pairs is equal to $|W^k|(n - |W^k|)$, obtained as $n(n-1)/2$ (total number of pairs of indices) minus $|W^k|(|W^k| - 1)/2$ (number of pairs of indices belonging to W^k) minus $(n - |W^k|)(n - |W^k| - 1)/2$ (number of pairs of indices not belonging to W^k). Therefore, the cost of computing $\varrho_i(x^{k+1})$ for all $i \in W^k$ and $\phi_i(x^{k+1})$ for all $i \notin W^k$ is

$$\mathcal{O}\big(|W^k|(n - |W^k|)\big).$$

The last cost in (14) is the one for computing $\phi_i(x^k)$ for all $i \notin W^k$. This cost is still the order of the number of pairs of indices such that exactly one of them belongs to W^k, that is, $\mathcal{O}\big(|W^k|(n - |W^k|)\big)$.

Summing all up, we get that computing ∇f as in (14) has a cost of the order of $|W^k|(|W^k| - 1)/2 + 2|W^k|(n - |W^k|)$, that is,

$$\mathcal{O}\left(\frac{|W^k|(4n - 3|W^k| - 1)}{2}\right). \tag{15}$$

Comparing (15) with (13), we can conclude that applying (14) is more convenient than (12) if $|W^k|(4n - 3|W^k| - 1) < n(n - 1)$. It is straightforward to verify that this inequality is satisfied if

$$|W^k| < \frac{n - 1}{3}.$$

So, in our implementation of FAST-ATVO, at each iteration we use either (12) or (14) to compute $\nabla f(x^k)$, depending on the dimension of $|W^k|$. Let us also point out that this complexity analysis shows that a decomposition approach, such as the one used in FAST-ATVO, can be particularly well suited for solving problem (8).

For what concerns the computation of the objective function, we may use similar arguments as above to show that a cost lower than $\mathcal{O}(n(n - 1)/2)$ can be obtained if $|W^k|$ is sufficiently small. Though, in our implementation of FAST-ATVO we use the following formula to compute the objective function:

$$f(x) = \frac{\nabla f(x)^T x}{p}. \tag{16}$$

Since we need to compute $\nabla f(x^k)$ at every iteration of FAST-ATVO, this choice seems particularly convenient as the cost of (16) is linear once the gradient is known. The only drawback is that, at a given iteration k, we may compute the objective function many times before producing the successive x^{k+1}: in this case we would compute many gradients that are not needed. But this can only occur if the unit stepsize is not accepted in the line search procedure (see Algorithm 4 in Appendix A), and our experiments showed that this is an unlikely event.

4   Experiments

In this section we apply our method to several real-world and established random benchmark networks with the aim of highlighting the improvements that the exact modularity total variation approach here proposed ensures over the standard linear method [45] and the nonlinear eigenvector method previously proposed in [58].

We subdivide the discussion in two parts: first, in Subsection 4.1 we provide details on the parameters used in our implementation of `FAST-ATVO`; then, in Subsection 4.2 we report extensive numerical results on sixteen datasets.

4.1   Implementation details

In this subsection, we provide details on the parameters used in our experiments to approximate problem (7) and to execute `FAST-ATVO`.

For what concerns problem (7), we used $p = 1.4$ to define the function $TV_Q^p(x)$. This choice of p has been guided by an extensive parameter–tuning phase and it has been chosen because it performed best overall on the datasets we have analyzed. As for the choice of the box constraints, we set $a = b = 1$ to have a more fair comparison with the Generalized RatioDCA approach, which was designed to solve that particular problem setting.

For what concerns `FAST-ATVO`, with reference to Algorithm 4 reported in Appendix A, we set $Z = 20$, $M = 100$, $\Delta_0 = 1\mathrm{e}20$, $\beta = 0.99$, $\delta = 0.5$ and $\gamma = 1\mathrm{e}{-3}$. The working set W^k is computed randomly at every iteration (except for the index $\hat{\imath}^k$ that is always included in W^k) and its dimension starts from 2 and is gradually increased through the iterations up to $\max(10, \min(1000, 0.03n))$. The search direction is computed as explained in Appendix A, using $\mu_{\min} = 1\mathrm{e}{-10}$ and $\mu_{\max} = 1\mathrm{e}10$.

Moreover, we use the following initialization strategy, that our preliminary experiments suggested to be useful for improving performances in practice: given any starting point x^0, all negative (positive) components of x^0 are set to the lower (upper) bound.

Finally, in the global optimization strategy described in Subsection 3.2 and Algorithm 3 we set $\sigma = 75$, which was found to give good results.

All the parameters used in the experiments have been chosen after a parameter–tuning phase performed over the analyzed data sets. So, in our experience, these parameters can be considered a good general setting.

4.2   Results

In this section we apply our method to several real-world and random benchmark networks with the aim of highlighting: (a) the improvements that `FAST-ATVO` ensures with respect to the previously proposed Generalized RatioDCA and (b) the overall improvement that the continuous modularity total variation approach we propose here ensures over the standard linear relaxation approach.

In particular, in our experiments we first tested the local optimization method `FAST-ATVO`, described in Subsection 3.1, and then the global optimization method `PS`, described in Subsection 3.2. We considered more than 30 datasets, but, for the sake of brevity, here we report results for 16 moderate-to-large size networks, ranging from around 7000 up to around 65000 nodes. The

Table 1: List of datasets used in the experiments, with corresponding sizes (in terms of number of nodes) and references.

Dataset ID	Dataset name	# nodes	Reference
1	geom	7343	[4]
2	as-735	7716	[41]
3	ca-HepTh	9877	[42]
4	vsp_c-30_data_data	11023	[52]
5	Oregon-1	11492	[41]
6	ca-HepPh	12008	[4]
7	vsp_befref_fxm_2_4_air02	14109	[52]
8	ca-AstroPh	18772	[42]
9	ca-CondMat	23133	[42]
10	rgg_n_2_15_s0	32768	[52]
11	vsp_sctap1-2b_and_seymourl	40174	[52]
12	vsp_model1_crew1_cr42_south31	45101	[52]
13	Words28	52652	[4]
14	vsp_bump2_e18_aa01_model1_crew1	56438	[52]
15	loc-Brightkite	58228	[13]
16	rgg_n_2_16_s0	65536	[52]

results we obained on the remaining datasets are aligned with the ones presented here. The list of the 16 datasets is shown in Table 1. All the datasets are publicly available and all the methods are implemented in C++. The software for implementing FAST-ATVO is available at

https://github.com/acristofari/fast-atvo

The networks we selected are primarily real networks, plus several randomly generated graphs borrowed from established benchmarks. In particular: geom is a collaboration network in computational geometry [4]; as-735 and Oregon-1 are snapshots of Internet at the level of autonomous systems [41]; ca-HepTh, ca-HepPh, ca-AstroPh and ca-CondMat are collaboration networks based on arXiv papers in High Energy Physics, Astro Physics and Condensed matter, respectively [42]; rgg_n_2_15_s0 and rgg_n_2_16_s0 are random geometric graph with 2^{15} (resp. 2^{16}) vertices used for the DIMACS10 challenge on graph clustering and partitioning [52]; Words28 is a dictionary graph built using a collection of close words in modern English [4]; vsp_c-30_data_data, vsp_befref_fxm_2_4_air02, vsp_sctap1-2b_and_seymourl, vsp_model1_crew1_cr42_south31 and vsp_bump2_e18_aa01_model1_crew1 are DIMACS10 star–like random benchmark graphs consisting of a mixture of social networks, finite-element graphs, VLSI chips, peer-to-peer networks and graphs from optimization solvers [52]; loc-Brightkite is a Location-based social network [13].

Performance results are shown in Tables 2 and 3 and Figure 1. Table 2 compares the modularity value obtained with FAST-ATVO, with Generalized RatioDCA and with the standard linear spectral method. The results shown in this table have been obtained using the leading eigenvector of the modularity matrix (what we call *linear eigenvector*) as a starting point of both FAST-ATVO and Generalized RatioDCA algorithms. From this table we can see that FAST-ATVO generally performs best with improvements in terms of modularity value up to 140% more than the linear eigenvector (for ca-HepTh dataset) and up to 97% more than the Generalized RatioDCA method (for vsp_befref_fxm_2_4_air02 dataset). Moreover, note that the communities identified by each method significantly different. This is also highlighted in Table 2, where we compare the sizes of such communities.

Table 2: Performance comparison using the linear eigenvector as starting point. The values shown here correspond to the value of $Q(S^*)$ and the size of S^*, where S^* is the community identified by optimal thresholding the output of the three methods: linear ($\mathbf{Q_{linear}}$ and $\mathbf{SIZE_{linear}}$, respectively), Generalized RatioDCA ($\mathbf{Q_R}$ and $\mathbf{SIZE_R}$, respectively) and FAST-ATVO ($\mathbf{Q_F}$ and $\mathbf{SIZE_F}$, respectively). In the columns corresponding to the size of S^* we also show, between round brackets, the percentage of nodes in S^* as compared with the whole set of nodes.

| Dataset ID	$\mathbf{Q_{linear}}$	$\mathbf{Q_R}$	$\mathbf{Q_F}$	$\mathbf{SIZE_{linear}}$	$\mathbf{SIZE_R}$	$\mathbf{SIZE_F}$	$\dfrac{\mathbf{Q_F}}{\mathbf{Q_{linear}}}$	$\dfrac{\mathbf{Q_F}}{\mathbf{Q_R}}$
1	0.28	0.29	0.40	3522 (48%)	3128 (43%)	2467 (34%)	1.46	1.40
2	0.20	0.25	0.37	3072 (40%)	3530 (46%)	3187 (41%)	1.89	1.51
3	0.17	0.39	0.40	1517 (15%)	3624 (37%)	3324 (34%)	2.40	1.03
4	0.33	0.41	0.47	5285 (48%)	3673 (33%)	4920 (45%)	1.42	1.17
5	0.28	0.31	0.39	4448 (39%)	5051 (44%)	5494 (48%)	1.38	1.25
6	0.35	0.36	0.41	1117 (9%)	971 (8%)	1166 (10%)	1.16	1.15
7	0.21	0.21	0.42	3346 (24%)	3346 (24%)	6738 (48%)	1.97	1.97
8	0.25	0.33	0.36	4186 (22%)	7448 (40%)	6334 (34%)	1.44	1.07
9	0.23	0.38	0.39	8105 (35%)	10801 (47%)	6307 (27%)	1.68	1.04
10	0.32	0.44	0.50	11613 (35%)	14228 (43%)	16333 (50%)	1.55	1.12
11	0.23	0.23	0.39	16754 (42%)	16754 (42%)	6314 (16%)	1.70	1.70
12	0.26	0.26	0.40	6176 (14%)	6176 (14%)	13438 (30%)	1.56	1.56
13	0.32	0.46	0.43	7565 (14%)	11538 (22%)	23620 (45%)	1.32	0.93
14	0.27	0.35	0.44	8289 (15%)	9013 (16%)	22045 (39%)	1.66	1.25
15	0.20	0.35	0.37	4495 (8%)	24931 (43%)	18081 (31%)	1.83	1.07
16	0.31	0.44	0.50	23054 (35%)	28106 (43%)	32600 (50%)	1.60	1.14

The stochastic nature of the working set selection combined with the non-monotone stabilization strategy we use to generate the new iterates are major differences between the proposed FAST-ATVO and the Generalized RatioDCA method. In our opinion, these two features allow the method to escape from "bad" local optima and quickly find much better modules. In fact, Generalized RatioDCA seems to often get trapped into a local optima somewhat near the initial linear eigenvector assignment, while FAST-ATVO is able to efficiently move further away. The phenomenon is particularly evident for the datasets $\{1, 6, 7, 11, 12\}$, as shown in Table 2, and it is further highlighted in the example drawing of Figure 2.

Table 3 shows mean and standard deviations of the modularity values obtained by FAST-ATVO and Generalized RatioDCA over 10 runs, each with a different randomly chosen starting point. This table confirms the behavior observed in Table 2: the average modularity value obtained with FAST-ATVO highly outperforms both the linear method and Generalized RatioDCA. Moreover, the modularity standard deviations shown in the table demonstrate that FAST-ATVO is generally more robust than Generalized RatioDCA.

The results of Tables 2 and 3 are summarized and shown together in the boxplots of Figure 1 (removing outliers), where we also compare the execution times of the methods. The upper panels in Subfigures 1a and 1b show medians and quartiles of the modularity values obtained with FAST-ATVO (left columns) and Generalized RatioDCA (right columns) over 10 runs with random starting points, together with the value obtained using the linear eigenvector as a starting point (yellow dot). A straight green line shows, instead, the modularity value obtained with the linear spectral method. The lower panels in the subfigures show median and quartiles of execution times of FAST-ATVO and Generalized RatioDCA. As for the modularity values, we can see that
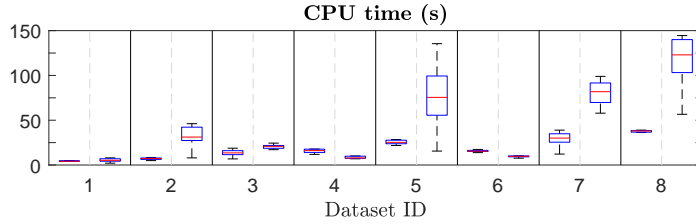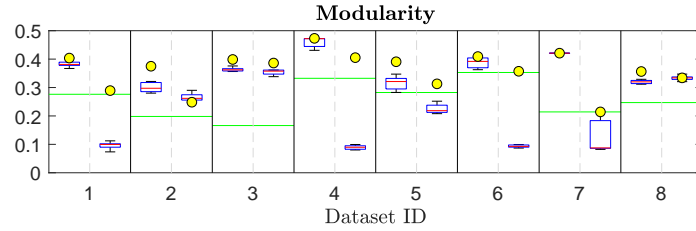
(a) Boxplots for graphs with less than 20000 nodes.
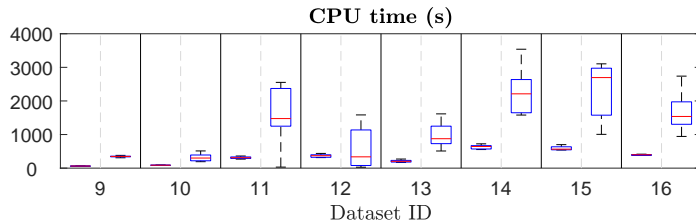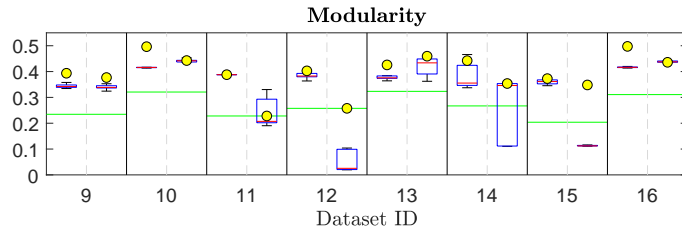
(b) Boxplots for graphs with more than 20000 nodes.

Figure 1: Boxplots of modularity values (upper panels) and execution times (lower panels) for `FAST-ATVO` (left columns) and for Generalized RatioDCA (right columns). Outliers were removed from the boxplots. The green lines are the values of modularity obtained using the linear method, whereas the yellow dots are the values of modularity obtained using the linear eigenvector as starting point.

`FAST-ATVO` is generally more efficient (as it requires a smaller median execution time) and more robust (as the variance of the CPU time is in general remarkably smaller).

Finally, we report in Figure 3 the performance comparison among `PS` framework, `FAST-ATVO` and Generalized RatioDCA. For the three methods we use the linear eigenvector as starting

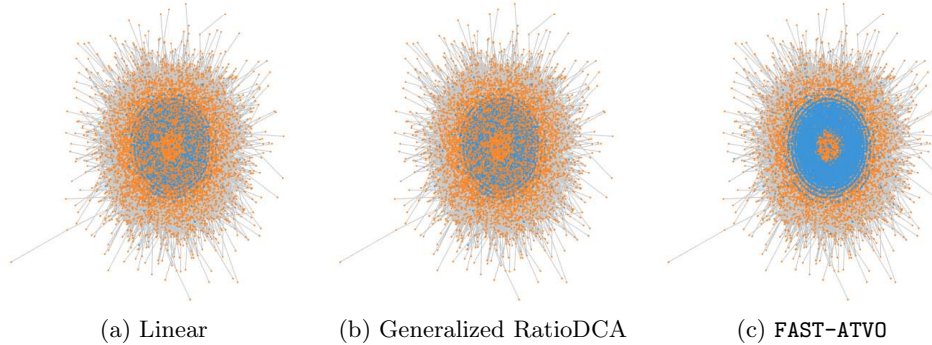|		
(a) Linear	(b) Generalized RatioDCA	(c) `FAST-ATVO`

Figure 2: Graph drawing of the leading module obtained on the dataset number 7 for the linear method (left), Generalized RatioDCA (center) and the proposed `FAST-ATVO` (right). In this case, Generalized RatioDCA returns a solution very close to the linear eigenvector assignment. This might indicate that Generalized RatioDCA gets trapped in the basin of attraction of a local minimum. `FAST-ATVO`, instead, is able to escape that point and to find a much better leading module assignment.

Table 3: Performance comparison using random starting points. The values shown here correspond to the mean and standard deviation of $Q(S^*)$, where S^* is the community identified by optimal thresholding the output of the three methods: linear ($\mathbf{Q_{linear}}$), Generalized RatioDCA ($\mathbf{Q_R}$) and `FAST-ATVO` ($\mathbf{Q_F}$).

Dataset ID	$\mathbf{Q_{linear}}$	avg $\mathbf{Q_R}$	std $\mathbf{Q_R}$	avg $\mathbf{Q_F}$	std $\mathbf{Q_F}$	$\dfrac{\text{avg } \mathbf{Q_F}}{\mathbf{Q_{linear}}}$	$\dfrac{\text{avg } \mathbf{Q_F}}{\text{avg } \mathbf{Q_R}}$
1	0.28	0.11	0.06	0.38	0.01	1.39	3.41
2	0.20	0.27	0.01	0.31	0.03	1.56	1.16
3	0.17	0.36	0.01	0.37	0.01	2.20	1.03
4	0.33	0.12	0.10	0.45	0.04	1.35	3.84
5	0.28	0.23	0.03	0.32	0.03	1.13	1.39
6	0.35	0.12	0.08	0.39	0.02	1.10	3.31
7	0.21	0.15	0.12	0.40	0.07	1.87	2.61
8	0.25	0.33	0.01	0.32	0.01	1.31	0.97
9	0.23	0.34	0.01	0.35	0.02	1.48	1.02
10	0.32	0.44	0.01	0.42	0.02	1.32	0.95
11	0.23	0.24	0.05	0.38	0.01	1.69	1.60
12	0.26	0.07	0.07	0.38	0.01	1.49	5.89
13	0.32	0.42	0.03	0.38	0.02	1.18	0.91
14	0.27	0.24	0.13	0.38	0.05	1.42	1.57
15	0.20	0.13	0.07	0.36	0.01	1.77	2.69
16	0.31	0.44	0.01	0.42	0.02	1.36	0.96

point. As we can easily see, the use of a global optimization strategy improves modularity values with respect to `FAST-ATVO`, while guaranteeing good performances in terms of CPU time. In particular, we notice that `PS` gives a modularity value higher than Generalized RatioDCA even for dataset `Words28`, which was the only one where `FAST-ATVO` was outperformed by Generalized RatioDCA.
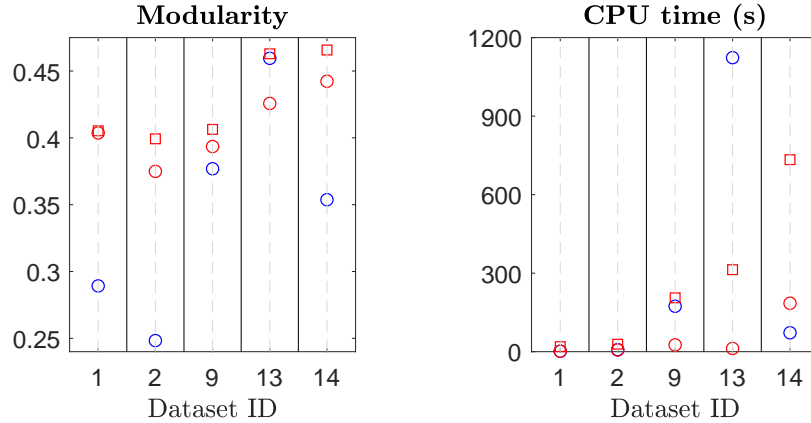
Figure 3: Modularity values (left panel) and execution times (right panel) for `PS` (red squares) `FAST-ATVO` (red circles) and Generalized RatioDCA (blue circles). Linear eigenvector is used as starting point for all methods.

5 Conclusions

In this paper, we described a new modularity total variation approach for the leading community problem. We first proved that the original combinatorial problem is equivalent to the minimization of a suitably chosen continuous nonsmooth function (the modularity total variation) over a box. We then considered a smooth approximation of the continuous problem and developed an algorithmic framework to efficiently tackle it. The reported results show that the proposed method outperforms both Generalized RatioDCA and the linear method in terms of modularity value. Furthermore, the CPU time required to find those good solutions is usually significantly smaller than the CPU time needed to run Generalized RatioDCA. Hence, even if the modularity landscape is usually fraught with local optima (see, e.g., [29]), the proposed approach quickly finds solutions with good modularity values in the end. As far as we can see, the use of both the non-monotone line search and the randomized working set selection seems to be of help in avoiding/escaping "bad" local optima.

Future work might focus on adapting our strategy to efficiently handle applications where dense weighted graphs (such as similarity or correlation graphs) or multilayer graphs are required. Another interesting research direction might be analyzing how the described approach transfers and applies to other community detection techniques like, e.g., Belief Propagation [59], Stochastic Block Models [1] and Infomap [51].

A Detailed scheme and convergence analysis of `FAST-ATVO`

In this appendix, we report the detailed scheme of `FAST-ATVO` (a short scheme was given in Algorithm 1), together with its convergence analysis. The detailed scheme is reported below in Algorithm 4.

We see that every iteration k starts with a non-stationary point x^k. We first compute the

Algorithm 4 FAST-ATVO(x^0) – detailed scheme

---

0 Given a feasible point x^0, fix $Z \geq 1$, $M \geq 0$, $\Delta_0 \geq 0$, $\beta \in (0,1)$, $\delta \in (0,1)$, $\gamma \in (0,1)$,
  $0 < \mu_{\min} \leq \mu_{\max} < \infty$ and set $k = 0$, $j = 0$, $l^0 = 0$, $f_R^0 = f^0 = f(x^0)$, $\Delta = \Delta_0$

1 While x^k is a non-stationary point for problem (8)

      Active and non-active set estimate
2   Compute $A_l^k = A_l(x^k)$, $A_u^k = A_u(x^k)$ and $N^k = N(x^k)$

      Function control every Z iterations
3   If $k = l^j + Z$, then compute $f(x^k)$
4   If $f(x^k) \geq f_R^j$
5   Backtrack to x^{l^j}, set $d^k = d^{l^j}$, $k = l^j$ and go to step 22
6   Else
7   Set $j = j + 1$, $l^j = k$, $f^j = f(x^k)$ and $f_R^j = \max\limits_{0 \leq i \leq \min\{j,M\}} f^{j-i}$
8   End if
9   End if

      Computation of the search direction
10   Find $\hat{\imath}^k \in \text{Argmax}_{i \in N^k} |x_i^k - [x^k - \nabla f(x^k)]_i^\sharp|$
11   Choose a non-empty working set $W^k \subseteq N^k$ such that $\hat{\imath}^k \in W^k$
12   Set $d_{A_l^k}^k = 0$, $d_{A_u^k}^k = 0$, $d_{N^k \setminus W^k}^k = 0$ and set $d_{W^k}^k = -\frac{1}{\mu^k}\nabla_{W^k} f(x^k)$, with $\mu^k \in [\mu_{\min}, \mu_{\max}]$

      Test for accepting the unit stepsize
13   If $\|[x^k + d^k]^\sharp\| \leq \Delta$
14   Set $x^{k+1} = [x^k + d^k]^\sharp$, $\Delta = \beta\Delta$, $k = k + 1$ and go to step 1
15   Else if $k \neq l^j + Z$, then compute $f(x^k)$
16   If $f(x^k) \geq f_R^j$
17   Backtrack to x^{l^j}, set $d^k = d^{l^j}$, $k = l^j$ and go to step 22
18   Else
19   Set $j = j + 1$, $l^j = k$, $f^j = f(x^k)$ and $f_R^j = \max\limits_{0 \leq i \leq \min\{j,M\}} f^{j-i}$
20   End if
21   End if

      Non-monotone Armijo line search
22   Set $\alpha^k = (\delta)^\nu$, where ν is the smallest non-negative integer such that

$$f([x^k + \delta^\nu d^k]^\sharp) \leq f_R^j + \gamma(\delta)^\nu \nabla f(x^k)^\top d^k$$

23   Set $x^{k+1} = [x^k + \alpha^k d^k]^\sharp$ and $k = k + 1$

24 End while

---

active and non-active sets estimates as in step 2 and, if necessary, we evaluate the objective function (steps 3–9). More in detail, thanks to a non-monotone stabilization strategy, inspired from that used in [30], we can compute $f(x^k)$ only once every Z iterations (if some tests described below are satisfied), rather than at each iteration, allowing us to save computational time. When we perform this function control, we compare $f(x^k)$ with a reference value f_R^j, which is the maximum among the last M function evaluations: if $f(x^k) \geq f_R^j$ we backtrack to the best point computed so far (i.e., x^{l^j}) and start a line search, otherwise we update f_R^j and go on.

21

At steps 10–12, we choose a non-empty working set $W^k \subseteq N^k$ by a Gauss-Southwell-type (or greedy) rule: W^k must contain the index $\hat{\imath}^k$ of the variable that most violates stationarity. We then compute a search direction d^k such that

$$d^k_{W^k} = -\frac{1}{\mu^k} \nabla_{W^k} f(x^k), \tag{17}$$

with $\mu^k > 0$, and all the other components of d^k are equal to zero. We see that, at each iteration, only a subset of variables (i.e., those in W^k) can be moved and the search direction is computed in the variable subspace defined by W^k. The computation of the coefficient μ^k will be described in details later on.

Afterwards, in steps 13–21, if $\|[x^k + d^k]^\sharp\|$ if sufficiently small we set $x^{k+1} = [x^k + d^k]^\sharp$ and terminate the iteration. It means that, if that test is satisfied, we accept the unit stepsize without computing the objective function. Otherwise, we have two possibilities: if $f(x^k) \geq f_R^j$ we backtrack to x^{l^j}, else we update f_R^j and go on. Both these cases are followed by a non-monotone line search.

In the last steps 22–23 of the algorithm, we compute the stepsize α^k in order to set $x^{k+1} = [x^k + \alpha^k d^k]^\sharp$. We use a non-monotone Armijo line search with reference value equal to f_R^j (which was first proposed in [30]).

Now, let us describe how we compute the search direction d^k (step 12). As mentioned before, we choose a coefficient μ^k (in a finite positive interval) and compute $d^k_{W^k}$ as in (17), while all the other components of d^k are set to zero. In this way, at each iteration k we move only the variables in W^k. In our experiments, we computed $d^k_{W^k}$ as a spectral (or Barzilai-Borwein) gradient direction (see, e.g., [26] and the references therein).

More precisely, based on the strategy proposed in [2, 6], for $k < 2$ we set

$$\mu^k = \max\left\{\mu_{\min}, \min\left\{1, \frac{\|x^k_{W^k}\|}{\|\nabla_{W^k} f(x^k)\|}\right\}\right\}$$

and, for $k \geq 2$,

$$\mu^k = \begin{cases} \max\{\mu_{\min}, \mu_a^k\}, & \text{if } 0 < \mu_a^k < \mu_{\max}, \\[2mm] \max\{\mu_{\min}, \min\{\mu_{\max}, \mu_b^k\}\}, & \text{if } \mu_a^k \geq \mu_{\max}, \\[2mm] \max\left\{\mu_{\min}, \min\left\{1, \dfrac{\|x^k_{W^k}\|}{\|\nabla_{W^k} f(x^k)\|}\right\}\right\}, & \text{if } \mu_a^k \leq 0, \end{cases}$$

with $0 < \mu_{\min} \leq \mu_{\max} < \infty$, $\mu_a^k = \dfrac{(s^{k-1})^\top y^{k-1}}{\|s^{k-1}\|^2}$, $\mu_b^k = \dfrac{\|y^{k-1}\|^2}{(s^{k-1})^\top y^{k-1}}$, $s^{k-1} = x^k_{W^k} - x^{k-1}_{W^k}$ and $y^{k-1} = \nabla_{W^k} f(x^k) - \nabla_{W^k} f(x^{k-1})$.

A.1   Global convergence proof

In order to prove the global convergence of `FAST-ATVO` we need some preliminary results that we derive below. Throughout the whole section, we use the notation of Algorithm 4.

First, we report a known result of non-monotone methods, whose proof can be easily adapted from the proof of Lemma 3 of [31] and is omitted here for the sake of brevity.

Lemma 4. *Assume that $\{x^k\}$ is an infinite sequence of points produced by* FAST-ATVO. *Then,*

$$\lim_{k \to \infty} f(x^k) = \lim_{j \to \infty} f_R^j = \bar{f}_R \in \mathbb{R}, \tag{18}$$

$$\lim_{k \to \infty} \|x^{k+1} - x^k\| = 0. \tag{19}$$

Further, we state the following technical lemma

Lemma 5. *There exist subsequences $\{\alpha^k\}_K$, $\{x^k\}_K$, $\{d^k\}_K$ and sets $\bar{A}_l$, $\bar{A}_u$, $\bar{N}$, $\bar{W}$ such that*

$$\lim_{k \to \infty, k \in K} \alpha^k = \bar{\alpha} \in \mathbb{R},$$

$$\lim_{k \to \infty, k \in K} x^k = \bar{x} \in \mathbb{R}^n,$$

$$\lim_{k \to \infty, k \in K} d^k = \bar{d} \in \mathbb{R}^n,$$

$$A_l^k = \bar{A}_l, \quad A_u^k = \bar{A}_u, \quad N^k = \bar{N}, \quad W^k = \bar{W}, \quad \forall k \in K,$$

with $\bar{\alpha} > 0$ and $\|\bar{d}\| > 0$.

Proof. The thesis is an immediate consequence of the fact that A_l^k, A_u^k, N^k, W^k are subsets of a finite set of indices and the sequences $\{\alpha^k\}$, $\{x^k\}$, $\{d^k\}$ are bounded. In particular, the latter property itself follows from the fact that $0 \leq \alpha^k \leq 1$, from the compactness of the feasible set and from the compactness of the feasible set combined with the definition of d^k and the continuity of ∇f, respectively. $\qquad \square$

Using the previous lemmas, we derive the following result.

Lemma 6. *Assume that $\{x^k\}$ is an infinite sequence of points produced by* FAST-ATVO. *Then,*

$$\lim_{k \to \infty} \alpha^k \|d^k\| = 0.$$

Proof. We proceed by contradiction and we assume that the result is not true. Let $\{\alpha^k\}_K$, $\{x^k\}_K$, $\{d^k\}_K$ and $\bar{A}_l$, $\bar{A}_u$, $\bar{N}$, $\bar{W}$ be as in Lemma 5. Using (19) and the continuity of the projection operator, we can thus write

$$\lim_{k \to \infty, k \in K} (x_i^{k+1} - x_i^k) = \lim_{k \to \infty, k \in K} ([\bar{x} + \bar{\alpha}\bar{d}]_i^\sharp - \bar{x}_i) = 0, \quad i = 1, \ldots, n. \tag{20}$$

In the sequel, we show that the above limit leads to $\|\bar{d}\| = 0$, getting a contradiction. To this extent, let us first observe that, from the definition of d^k and the continuity of ∇f, we have

$$\bar{d}_{\bar{W}} = -\frac{1}{\bar{\mu}} \nabla_{\bar{W}} f(\bar{x}), \tag{21}$$

for some $\bar{\mu} > 0$. Moreover, from our estimates (10) and the continuity of ∇f, there exists an iteration $\hat{k}$ such that

$$\bar{x}_i = l_i, \nabla_i f(\bar{x}) > 0 \quad \Rightarrow \quad i \notin \bar{N}, \forall k \geq \hat{k}, k \in K, \tag{22a}$$

$$\bar{x}_i = u_i, \nabla_i f(\bar{x}) < 0 \quad \Rightarrow \quad i \notin \bar{N}, \forall k \geq \hat{k}, k \in K. \tag{22b}$$

Now, let us consider any index $i \in \{1, \ldots, n\}$. We can distinguish four possible cases.

(i) $i \notin \bar{W}$. From the definition of the search direction, it follows that $\bar{d}_i = 0$.

(ii) $i \in \bar{W}$ such that $l_i < \bar{x}_i < u_i$. From (20) and the fact that $\bar{\alpha} > 0$, we obtain $\bar{d}_i = 0$.

(iii) $i \in \bar{W}$ such that $\bar{x}_i = l_i$. Recalling that $\bar{W} \subseteq \bar{N}$, from (22a) and the definition of d^k we have $\nabla_i f(\bar{x}) \leq 0$. Using (21) we obtain $\bar{d}_i \geq 0$, which, combined with (20) and the fact that $\bar{\alpha} > 0$, implies that $\bar{d}_i = 0$.

(iv) $i \in \bar{W}$ such that $\bar{x}_i = u_i$. Reasoning as above, we get $\bar{d}_i = 0$.

We thus obtain $\|\bar{d}\| = 0$, leading to a contradiction. $\qquad\square$

We can now show that the sequence of directional derivatives $\{\nabla f(x^k)^\top d^k\}$ converges to zero, which will be crucial to prove the global convergence of the algorithm.

Proposition 2. *Assume that $\{x^k\}$ is an infinite sequence of points produced by* FAST-ATVO. *Then,*

$$\lim_{k \to \infty} \nabla f(x^k)^\top d^k = 0. \tag{23}$$

Proof. By contradiction, we assume that (23) does not hold. There must exist $\{\alpha^k\}_K$, $\{x^k\}_K$, $\{d^k\}_K$ and $\bar{A}_l$, $\bar{A}_u$, $\bar{N}$, $\bar{W}$ defined as in Lemma 5, which also satisfy

$$\lim_{k \to \infty, \, k \in K} \nabla f(\bar{x})^\top \bar{d} = -\eta < 0. \tag{24}$$

Combining (24) with Lemma 6, we obtain

$$\lim_{k \to \infty, \, k \in K} \alpha^k = 0. \tag{25}$$

Therefore, there exist two further infinite subsequences, that, with a slight abuse of notation, we still denote by $\{x^k\}_K$ and $\{d^k\}_K$, such that $\alpha^k < 1$ for all $k \in K$. From Algorithm 4, α^k can be less than 1 if and only if it is computed by the non-monotone Armijo line search using $\nu \geq 1$ (see steps 22–23). It follows that

$$f\left(\left[x^k + \frac{\alpha^k}{\delta} d^k\right]^\sharp\right) > f_R^{q(k)} + \gamma \frac{\alpha^k}{\delta} \nabla f(x^k)^T d^k$$
$$\geq f(x^k) + \gamma \frac{\alpha^k}{\delta} \nabla f(x^k)^T d^k, \quad \forall k \in K, \tag{26}$$

where $q(k) = \max\{j : l^j \leq k\}$. Let us write the point $[x^k + \frac{\alpha^k}{\delta} d^k]^\sharp$ as follows:

$$\left[x^k + \frac{\alpha^k}{\delta} d^k\right]^\sharp = x^k + \frac{\alpha^k}{\delta} d^k - y^k, \tag{27}$$

where

$$y_i^k = \begin{cases} x_i^k + \dfrac{\alpha^k}{\delta} d_i^k - l_i, & \text{if } x_i^k + \dfrac{\alpha^k}{\delta} d_i^k < l_i \\[2mm] x_i^k + \dfrac{\alpha^k}{\delta} d_i^k - u_i, & \text{if } x_i^k + \dfrac{\alpha^k}{\delta} d_i^k > u_i \\[2mm] 0, & \text{otherwise,} \end{cases} \tag{28}$$

or equivalently,

$$y_i^k = \max\{0, (x^k + \frac{\alpha^k}{\delta}d^k)_i - u_i\} - \max\{0, l_i - (x^k + \frac{\alpha^k}{\delta}d^k)_i\}, \quad i = 1, \ldots, n.$$

Using (25), the feasibility of points x^k and the fact that $\{d^k\}$ is bounded, we have

$$\lim_{k \to \infty, \, k \in K} y^k = 0. \tag{29}$$

Now, from (26) and (27) we can write

$$f\left(x^k + \frac{\alpha^k}{\delta}d^k - y^k\right) - f(x^k) > \gamma\frac{\alpha^k}{\delta}\nabla f(x^k)^\top d^k, \quad \forall k \in K. \tag{30}$$

By the mean value theorem, we also have

$$f\left(x^k + \frac{\alpha^k}{\delta}d^k - y^k\right) = f(x^k) + \frac{\alpha^k}{\delta}\nabla f(z^k)^\top d^k - \nabla f(z^k)^\top y^k, \tag{31}$$

where $z^k = x^k + \theta^k(\alpha^k d^k/\delta - y^k)$ and $\theta^k \in (0,1)$. Using (25), (29) and the fact that $\{d^k\}$ is bounded, we obtain

$$\lim_{k \to \infty, \, k \in K} z^k = \bar{x}. \tag{32}$$

Moreover, from (30) and (31), we have

$$\nabla f(z^k)^\top d^k - \frac{\delta}{\alpha^k}\nabla f(z^k)^\top y^k > \gamma\nabla f(x^k)^\top d^k, \quad \forall k \in K. \tag{33}$$

In the following, we show that

$$\liminf_{k \to \infty, \, k \in K} \frac{\delta}{\alpha^k}\nabla f(z^k)^\top y^k \geq 0. \tag{34}$$

From (28) and the fact that each x^k is feasible, we first observe that

$$y_i^k \begin{cases} \in [\alpha^k d_i^k/\delta, 0], & \text{if } d_i^k < 0, \\ \in [0, \alpha^k d_i^k/\delta], & \text{if } d_i^k > 0, \\ = 0, & \text{if } d_i^k = 0. \end{cases} \tag{35}$$

Now, we consider any index $i \in \{1, \ldots, n\}$ and we analyze four possible cases, in order to show that (34) holds.

(i) $i \in \{1, \ldots, n\} \setminus \bar{W}$. From the rule used to compute d^k, we have $d_i^k = 0$. Using (35) it follows that $y_i^k = 0$ for all $k \in K$, implying that

$$\lim_{k \to \infty, \, k \in K} \frac{\delta}{\alpha^k}\nabla_i f(z^k)y_i^k = 0. \tag{36}$$

25

(ii) $i \in \bar{W}$ such that $l_i < \bar{x}_i < u_i$. Since $\{x^k\}_K \to \bar{x}$, for all sufficiently large $k \in K$ we have $l_i + \tau \le x^k \le u_i - \tau$ for some $\tau > 0$. Using (25) and the fact that $\{d^k\}$ is bounded, for all sufficiently large $k \in K$ we have

$$l_i < x_i^k + \frac{\alpha^k}{\delta} d_i^k < u_i,$$

which, combined with (28), implies that $y_i^k = 0$ for all sufficiently large $k \in K$. Then,

$$\lim_{k \to \infty, \, k \in K} \frac{\delta}{\alpha^k} \nabla_i f(z^k) y_i^k = 0. \tag{37}$$

(iii) $i \in \bar{W}$ such that $\bar{x}_i = l_i$. Since $\{x^k\}_K \to \bar{x}$ and ∇f is continuous, from our estimates (10) (and recalling that $\bar{W} \subseteq \bar{N}$) we have

$$\nabla_i f(\bar{x}) \le 0. \tag{38}$$

Now, we also show that there exists $\hat{k} \in K$ such that

$$y_i^k \le 0, \quad \forall k \ge \hat{k}, \, k \in K. \tag{39}$$

Indeed, since $\{x^k\}_K \to \bar{x}$, for all sufficiently large $k \in K$ we have $x^k \le u_i - \tau$ for some $\tau > 0$. Using (25) and the fact that $\{d^k\}$ is bounded, for all sufficiently large $k \in K$ we have

$$x_i^k + \frac{\alpha^k}{\delta} d_i^k < u_i,$$

which, combined with (28), implies (39). Let us partition K into K_1 and K_2, such that every $k \in K$ belongs to K_1 if and only if $d_i^k \ge 0$ (and then, every $k \in K$ belongs to K_2 if and only if $d_i^k < 0$). Assuming without loss of generality that both K_1 and K_2 are infinite, we now analyze the corresponding subsequences.

- For what concerns K_1, using (35) we have that $y_i^k \ge 0$ for all $k \in K_1$. From (39) it follows that $y_i^k = 0$ for all sufficiently large $k \in K_1$. Then,

$$\lim_{k \to \infty, \, k \in K_1} \frac{\delta}{\alpha^k} \nabla_i f(z^k) y_i^k = 0. \tag{40}$$

- For what concerns K_2, taking into account (38) we distinguish two possible cases.
  (a) $\nabla_i f(\bar{x}) < 0$. Since $\{z^k\}_K \to \bar{x}$, then $\nabla_i f(z^k) < 0$ for all sufficiently large $k \in K_2$. From (39) it follows that $\nabla f_i(z^k) y_i^k \ge 0$ for sufficiently large $k \in K_2$. Then,

$$\liminf_{k \to \infty, \, k \in K_2} \frac{\delta}{\alpha^k} \nabla_i f(z^k) y_i^k \ge 0. \tag{41}$$

  (b) $\nabla_i f(\bar{x}) = 0$. From (35) we have $|y_i^k| \le \frac{\alpha^k}{\delta} |d_i^k|$. Therefore,

$$0 < \frac{\delta}{\alpha^k} |\nabla_i f(z^k) y_i^k| \le \frac{\delta}{\alpha^k} |\nabla_i f_i(z^k)| |y_i^k| \le |\nabla_i f(z^k)| |d_i^k|.$$

  Since $\{z^k\}_K \to \bar{x}$ and $\{d^k\}$ is bounded, from the continuity of ∇f we get

$$\lim_{k \to \infty, \, k \in K_2} \frac{\delta}{\alpha^k} \nabla_i f(z^k) d_i^k = 0. \tag{42}$$

26

From (40),(41) and (42) we obtain that, for all $i \in \bar{W}$ such that $\bar{x}_i = l_i$,

$$\liminf_{k \to \infty, \, k \in K} \frac{\delta}{\alpha^k} \nabla_i f(z^k) y_i^k \geq 0. \tag{43}$$

(iv) $i \in \hat{N}$ such that $\bar{x}_i = u_i$. Reasoning as in the previous case, we obtain

$$\liminf_{k \to \infty, \, k \in K} \frac{\delta}{\alpha^k} \nabla_i f(z^k) y_i^k \geq 0. \tag{44}$$

Therefore, (34) follows from (36), (37), (43) and (44). Combining (34) with (33), we can write

$$\begin{aligned}
0 &\leq \liminf_{k \to \infty, \, k \in K} \left(\nabla f(z^k)^\top d^k - \frac{\delta}{\alpha^k} \nabla f(z^k)^\top y^k - \gamma \nabla f(x^k)^\top d^k \right) \\
&\leq \liminf_{k \to \infty, \, k \in K} \left(\nabla f(z^k)^\top d^k - \gamma \nabla f(x^k)^\top d^k \right) \\
&= \lim_{k \to \infty, \, k \in K} \left(\nabla f(z^k)^\top d^k - \gamma \nabla f(x^k)^\top d^k \right) = (1 - \gamma) \nabla f(\bar{x})^\top \bar{d},
\end{aligned}$$

where the last two equalities follow from the continuity of ∇f and the fact that both $\{x^k\}_K$ and $\{z^k\}_K$ converge to $\bar{x}$. Using (24), we finally obtain $(\gamma - 1)\eta \geq 0$, with $\eta > 0$ and $\gamma \in (0, 1)$, which leads to a contradiction. $\square$

We are finally ready to prove the global convergence of `FAST-ATVO` to stationary points.

Proof of Theorem 2. Assume that the sequence $\{x^k\}$ generated by `FAST-ATVO` is infinite and let x^* be a limit point of $\{x^k\}$. Further, let $\{\alpha^k\}_K$, $\{x^k\}_K$, $\{d^k\}_K$ and $\bar{A}_l$, $\bar{A}_u$, $\hat{N}$, $\bar{W}$ be as in Lemma 5. Using the stationarity conditions (9), we can measure the stationarity violation of any feasible point x by the following functions:

$$\phi(x_i) = \min\left\{ \max\{l_i - x_i, -\nabla_i f(x)\}^2, \max\{x_i - u_i, \nabla_i f(x)\}^2 \right\}, \quad i = 1, \ldots, n.$$

Namely, a feasible point x is stationary if and only if $\phi(x_i) = 0$ for all $i = 1, \ldots, n$. Now, arguing by contradiction, assume that x^* is non-stationary. Then, an index i such that $\phi(x_i^*) > 0$ exists. Note that it must hold that

$$\nabla_i f(x^*) \neq 0. \tag{45}$$

Moreover, from the continuity of ϕ, there exist $\epsilon \in (0, 1)$ and $\hat{k} \in K$ such that

$$\phi(x_i^k) \geq \epsilon, \quad \forall k \geq \hat{k}. \tag{46}$$

Now, we consider four possible cases.

(i) $i \in \bar{A}_l$. From (10a) we have $x_i^k = l_i$ and $\nabla_i f(x^k) > 0$ for all $k \in K$. Since $\{x^k\}_K \to x^*$, from the continuity of ∇f it follows that, for all sufficiently large $k \in K$,

$$\nabla_i f(x^k) \geq -\frac{\epsilon}{2}.$$

Then, we have $\phi(x_i^k) \leq \epsilon^2/4 < \epsilon$ for all sufficiently large $k \in K$. This contradicts (46).

(ii) $i \in \bar{A}_u$. From (10b) we have $x_i^k = u_i$ and $\nabla_i f(x^k) < 0$ for all $k \in K$. Then, we obtain a contradiction by the same arguments used above.

(iii) $i \in \bar{W}$. Using Proposition (2), we have $\nabla f(x^*)^\top \bar{d} = 0$. From the definition of d^k and the continuity of ∇f, we obtain

$$0 = \nabla f(x^*)^\top \bar{d} = \nabla_{\bar{W}} f(x^*)^\top \bar{d}_{\bar{W}} = -\frac{1}{\bar{\mu}} \|\nabla_{\bar{W}} f(x^*)\|^2,$$

and then $\nabla_i f(x^*) = 0$, contradicting (45).

(iv) $i \in \bar{N} \setminus \bar{W}$. Without loss of generality, we can assume that the index $\hat{\imath}^k$ computed at step 10 is constant and equal to $\hat{\imath}$ for all $k \in K$ (passing into a subsequence if necessary). So, using the definition of $\hat{\imath}^k$, for all $k \in K$ we can write

$$|x_i^k - [x^k - \nabla f(x^k)]_i^\sharp| \leq |x_{\hat{\imath}^k}^k - [x^k - \nabla f(x^k)]_{\hat{\imath}^k}^\sharp| = |x_{\hat{\imath}}^k - [x^k - \nabla f(x^k)]_{\hat{\imath}}^\sharp|.$$

Since $x_{\hat{\imath}}^*$ does not violate stationarity (from the fact that $\hat{\imath} \in \bar{W}$, as analyzed above), the continuity of the projection operator and the continuity of ∇f imply that

$$0 \leq |x_i^* - [x^* - \nabla f(x^*)]_i^\sharp| = \lim_{k \to \infty, \, k \in K} |x_i^k - [x^k - \nabla f(x^k)]_i^\sharp|$$
$$\leq \lim_{k \to \infty, \, k \in K} |x_{\hat{\imath}}^k - [x^k - \nabla f(x^k)]_{\hat{\imath}}^\sharp|$$
$$= |x_{\hat{\imath}}^* - [x^* - \nabla f(x^*)]_{\hat{\imath}}^\sharp| = 0.$$

We hence have $\phi(x_i^*) = 0$, which gives a contradiction.

Therefore x^* must be a stationary point. $\qquad \square$

References

[1] E. Abbe. Community detection and stochastic block models: recent developments. *The Journal of Machine Learning Research*, 18(1):6446–6531, 2017.

[2] R. Andreani, E. G. Birgin, J. M. Martínez, and M. L. Schuverdt. Second-order negative-curvature methods for box-constrained and general constrained optimization. *Comput. Optim. Appl.*, 45(2):209–236, 2010.

[3] F. Bach. Learning with submodular functions: A convex optimization perspective. *Foundations and Trends in Machine Learning*, 6(2-3):145–373, 2013. ISSN 1935-8237.

[4] V. Batagelj and A. Mrvar. Pajek datasets collection, 2006. URL http://vlado.fmf.uni-lj.si/pub/networks/data/.

[5] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.*, 2:183–202, 2009.

[6] E. G. Birgin and J. M. Martínez. Large-scale active-set box-constrained optimization method with spectral projected gradients. *Comput. Optim. Appl.*, 23(1):101–125, 2002.

[7] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.*, 2008:P10008, 2008.

[8] Z. M. Boyd, E. Bae, X.-C. Tai, and A. L. Bertozzi. Simplified energy landscape for modularity using total variation. *SIAM J. Appl. Math.*, 78(5):2439–2464, 2018.

[9] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner. On finding graph clusterings with maximum modularity. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 121–132. Springer, 2007.

[10] X. Bresson, T. Laurent, D. Uminsky, and J. Von Brecht. Multiclass total variation clustering. In *Adv. Neural Inf. Process. Syst. (NIPS)*, pages 1421–1429, 2013.

[11] T. Bühler and M. Hein. Spectral clustering based on the graph *p*-Laplacian. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, pages 81–88, 2009.

[12] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *J. Math. Imaging Vision*, 40:120–145, 2011.

[13] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: User movement in location-based social networks. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2011.

[14] F. R. K. Chung and L. Lu. *Complex graphs and networks*, volume 107. American Mathematical Society, Providence, 2006.

[15] M. Cinelli, L. Peel, A. Iovanella, and J.-C. Delvenne. Network constraints on the mixing patterns of binary node metadata. *arXiv preprint arXiv:1908.04588*, 2019.

[16] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70:066111, 2004.

[17] A. Cristofari, M. De Santis, S. Lucidi, and F. Rinaldi. A two-stage active-set algorithm for bound-constrained optimization. *J. Optim. Theory Appl.*, 172(2):369–401, 2017.

[18] F. E. Curtis, Z. Han, and D. P. Robinson. A globally convergent primal-dual active-set framework for large-scale convex quadratic optimization. *Comput. Optim. Appl.*, 60(2):311–341, 2015.

[19] M. De Santis, G. Di Pillo, and S. Lucidi. An active set feasible method for large-scale minimization problems with bound constraints. *Comput. Optim. Appl.*, 53(2):395–423, 2012.

[20] G. Di Pillo and L. Grippo. A class of continuously differentiable exact penalty function algorithms for nonlinear programming problems. In *System Modelling and Optimization*, pages 246–256. Springer, 1984.

[21] J. Duch and A. Arenas. Community detection in complex networks using extremal optimization. *Phys. Rev. E*, 72:027104, 2005.

[22] F. Facchinei and S. Lucidi. Quadratically and superlinearly convergent algorithms for the solution of inequality constrained minimization problems. *J. Optim. Theory Appl.*, 85(2):265–289, 1995.

[23] D. Fasino and F. Tudisco. An algebraic analysis of the graph modularity. *SIAM J. Matrix Anal. Appl*, 35:997–1018, 2014.

[24] D. Fasino and F. Tudisco. Generalized modularity matrices. *Linear Algebra Appl.*, 502:327–345, 2016.

[25] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl. qpoases: A parametric active-set algorithm for quadratic programming. *Math. Program. Comput.*, 6(4):327–363, 2014.

[26] R. Fletcher. On the Barzilai-Borwein method. In *Optimization and control with applications*,

pages 235–256. Springer, 2005.

[27] S. Fortunato. Community detection in graphs. *Phys. Rep.*, 486:75–174, 2010.

[28] S. Fortunato and D. Hric. Community detection in networks: A user guide. *Phys. Rep.*, 659: 1–44, 2016.

[29] B. H. Good, Y.-A. de Montjoye, and A. Clauset. Performance of modularity maximization in practical contexts. *Phys. Rev. E*, 81(4):046106, 2010.

[30] L. Grippo, F. Lampariello, and S. Lucidi. A nonmonotone line search technique for newton's method. *SIAM J. Numer. Anal.*, 23(4):707–716, 1986.

[31] L. Grippo, F. Lampariello, and S. Lucidi. A class of nonmonotone stabilization methods in unconstrained optimization. *Numer. Math.*, 59(1):779–805, 1991.

[32] A. Grosso, M. Locatelli, and F. Schoen. A population-based approach for hard global optimization problems based on dissimilarity measures. *Math. Program.*, 110(2):373–404, 2007.

[33] R. Guimera, M. Sales-Pardo, and L. A. N. Amaral. Modularity from fluctuations in random graphs and complex networks. *Phys. Rev. E*, 70:025101, 2004.

[34] W. W. Hager and H. Zhang. A new active set algorithm for box constrained optimization. *SIAM J. Optim.*, 17(2):526–557, 2006.

[35] L. H. Hartwell, J. J. Hopfield, S. Leibler, and A. W. Murray. From molecular to modular cell biology. *Nature*, 402(6761supp):C47, 1999.

[36] M. Hein and T. Bühler. An inverse power method for nonlinear eigenproblems with applications in 1-spectral clustering and sparse PCA. In *Adv. Neural Inf. Process. Syst. 23 (NIPS)*, pages 847–855, 2010.

[37] M. Hein and S. Setzer. Beyond spectral clustering-tight relaxations of balanced graph cuts. In *Adv. Neural Inf. Process. Syst. (NIPS)*, pages 2366–2374, 2011.

[38] H. Hu, T. Laurent, M. A. Porter, and A. L. Bertozzi. A method based on total variation for network modularity optimization using the MBO scheme. *SIAM J. Appl. Math.*, 73: 2224–2246, 2013.

[39] A. Lancichinetti and S. Fortunato. Community detection algorithms: a comparative analysis. *Phys. Rev. E*, 80:056117, 2009.

[40] R. H. Leary. Global optimization on funneling landscapes. *J. Global Optim.*, 18(4):367–383, 2000.

[41] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2005.

[42] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. In *ACM Trans. Knowledge Discovery from Data*, 2007.

[43] P. Mercado, F. Tudisco, and M. Hein. Clustering signed networks with the geometric mean of Laplacians. In *Adv. Neural Inf. Process. Syst.(NIPS)*, 2016.

[44] P. Mercado, A. Gautier, F. Tudisco, and M. Hein. The power mean Laplacian for multilayer graph clustering. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 84 of *Proceedings of Machine Learning Research*, pages 1828–1838, 2018.

[45] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, 74:036104, 2006.

[46] M. E. J. Newman. *Networks: an introduction*. Oxford University Press, 2010.

[47] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, 2004.

[48] J. Nocedal and S. J. Wright. *Sequential quadratic programming*. Springer, 2006.

[49] M. Ozer, N. Kim, and H. Davulcu. Community detection in political twitter networks using nonnegative matrix factorization methods. In *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 81–88. IEEE Press, 2016.

[50] M. A. Porter, J.-P. Onnela, and P. J. Mucha. Communities in networks. *Notices Amer. Math. Soc.*, 56:1082–1097, 2009.

[51] M. Rosvall, D. Axelsson, and C. T. Bergstrom. The map equation. *The European Physical Journal Special Topics*, 178(1):13–23, 2009.

[52] I. Safro, P. Sanders, and C. Schulz. Advanced coarsening schemes for graph partitioning. *Journal of Experimental Algorithmics (JEA)*, 19:1–24, 2015.

[53] S. E. Schaeffer. Graph clustering. *Comput. Sci. Rev.*, 1:27–64, 2007.

[54] H.-W. Shen and X.-Q. Cheng. Spectral methods for the detection of network community structure: a comparative analysis. *J. Stat. Mech. Theory Exp.*, 2010:P10020, 2010.

[55] V. A. Traag, P. Van Dooren, and Y. Nesterov. Narrow scope for resolution-limit-free community detection. *Phys. Rev. E*, 84:016114, 2011.

[56] F. Tudisco and M. Hein. A nodal domain theorem and a higher-order Cheeger inequality for the graph p-Laplacian. *J. Spectr. Theory*, 8:883–908, 2018.

[57] F. Tudisco and D. J. Higham. A nonlinear spectral method for core-periphery detection in networks. *SIAM J. Mathematics of Data Science*, in press.

[58] F. Tudisco, P. Mercado, and M. Hein. Community detection in networks via nonlinear modularity eigenvectors. *SIAM J. Appl. Math.*, 78:2393–2419, 2018.

[59] P. Zhang and C. Moore. Scalable detection of statistically significant communities and hierarchies, using message passing for modularity. *Proceedings of the National Academy of Sciences*, 111(51):18144–18149, 2014.