Received: 16 October 2018 | Revised: 11 January 2019 | Accepted: 3 March 2019

DOI: 10.1002/cmm4.1020

RESEARCH ARTICLE

WILEY

Bootstrap AMG for spectral clustering

Pasqua D'Ambra[1] | Luisa Cutillo[2] | Panayot S. Vassilevski[3,4]

[1]Institute for Applied Computing, Consiglio Nazionale delle Ricerche, Napoli, Italy

[2]School of Mathematics, University of Leeds, Leeds, UK

[3]Fariborz Maseeh Department of Mathematics and Statistics, Portland State University, Portland, Oregon

[4]Center for Applied Scientific Computing, Lawrence Livermore National Laboratories, Livermore, California

Correspondence
Pasqua D'Ambra, Institute for Applied Computing, Consiglio Nazionale delle Ricerche, Via P. Castellino, 111, 80131 Napoli, Italy.
Email: pasqua.dambra@cnr.it

Funding information
Horizon 2020 Project by the European Council, Grant/Award Number: 676629; National Science Foundation, Grant/Award Number: DMS-1619640; Lawrence Livermore National Laboratory, U.S. Department of Energy, Grant/Award Number: DE-AC52-07NA27344

Graph Laplacian is a popular tool for analyzing graphs, particularly in graph partitioning and clustering. Given a notion of similarity (via an adjacency matrix), graph clustering refers to identifying different groups such that vertices in the same group are more similar compared to vertices across different groups. Data clustering can be reformulated in terms of a graph clustering problem when the given set of data is represented as a graph, also known as similarity graph. In this context, eigenvectors of the graph Laplacian are often used to obtain a new geometric representation of the original data set that generally enhances cluster properties and improves cluster detection. In this work, we apply a bootstrap algebraic multigrid (AMG) method that constructs a set of vectors associated with the graph Laplacian. These vectors, referred to as algebraically smooth ones, span a low-dimensional Euclidean space, which we use to represent the data, enabling cluster detection both in synthetic and in realistic well-clustered graphs. We show that, in the case of a good quality bootstrap AMG, the computed smooth vectors employed in the construction of the final AMG operator, which by construction is spectrally equivalent to the originally given graph Laplacian, accurately approximate the space in the lower portion of the spectrum of the preconditioned operator. Thus, our approach can be viewed as a spectral clustering technique associated with the generalized spectral problem (Laplace operator versus the final AMG operator), and hence, it can be seen as an extension of the classical spectral clustering that employs a standard eigenvalue problem.

KEYWORDS
algebraically smooth vectors, bootstrap AMG, graph Laplacian, spectral clustering

1 | INTRODUCTION

Our capability to understand complex systems in many areas, such as biology, social science, medicine, and technology, is largely based on the science of networks and their representation in terms of graphs. Let $X = \{x_1, \dots, x_n\}$ and $W = (w_{ij})_{i,j=1,\dots,n}$, be a set of data and a matrix of nonnegative entries corresponding to some measure of similarity between pairs of data, respectively. A way to represent the above set of data is the similarity graph, that is, a weighted undirected graph $G = (V, E, W)$, where the vertex set V consists of indices of the given data set, the edge set E corresponds to connected data so that $(i,j) \in E$ iff $w_{ij} > 0$, and W is the edge weight matrix. The cardinality or size of G is the dimension $|V| = n$ of V. A very popular and powerful tool for studying the graph G is its Laplacian matrix, also known as graph Laplacian, that is, $L = D - W \in \mathcal{R}^{n\times n}$, where $D = \text{diag}(d_i)_{i=1,\dots,n}$, with $d_i = \sum_{j=1}^{n} w_{ij}$, is the diagonal matrix of weighted vertex degrees. We observe that L is a symmetric, positive semidefinite M-matrix, and its spectrum is a valuable tool for studying the graph, eg, we note that $\mathrm{rank}(L) = n - c(G)$, where $c(G)$ is the number of connected components of G.[1]

Real-life networks, such as social networks, biochemical networks, and information networks, generally have well-expressed community structure, ie, they have a regular structure where we can identify groups or *clusters of vertices* with many edges among vertices inside the group and only few connections among vertices in different groups. This feature and the capability of automatically detecting such groups have important implications for our deep understanding of the networks. Given a graph partition $V_1, \dots, V_K$, that is, K nonempty sets $V_1, \dots, V_K$, so that $V_k \subset V$, $V_k \cap V_l = \emptyset, \forall k \neq l$ and $V_1 \cup \cdots \cup V_K = V$, let $W(V_k, \overline{V_k}) = \sum_{i \in V_k, j \in \overline{V_k}} w_{ij}$ be, where $\overline{V_k}$ is the complement of V_k in V. It is well-known that a graph partition, which minimizes the following *edge-cut* functional:

$$RatioCut(V_1, \dots, V_K) = \frac{1}{2} \sum_{k=1}^{K} \frac{W(V_k, \overline{V_k})}{|V_k|}, \tag{1}$$

can be obtained by computing the first K eigenvectors corresponding to the first K smallest eigenvalues of L.[2] We observe that a graph partition that minimizes (1) corresponds to a partition that minimizes the weight of the edges between two different sets and maximizes the number of the vertices within a set. The above problem is known as the *min-cut problem* in graph theory and is a widely used formulation for spectral clustering exploiting the eigenspace of graph Laplacian as low-dimensional geometric space for graph embedding and analysis. There are many approaches to clustering, and for exhaustive reviews of methods and applications, we refer the reader to the works of Schaeffer,[3] Fortunato,[4] and Nascimento and de Carvalho.[5]

In order to avoid some technical details with the semidefiniteness of L, in what follows, we consider the graph Laplacian of a connected graph and eliminate singularity by a rank-1 update of the matrix $L_S = L + \lambda\, \mathbf{e} \cdot \mathbf{e}^T$, where $\mathbf{e}$ is a vector of dimension n having nonzero components (unit values) only for the pair of indices i and j corresponding to a single arbitrary edge $(i,j) \in E$. Note that, in the case of graphs with more than one connected component, we apply our method to each of its connected components.

In this work, we propose to use as a new space for graph embedding, the space spanned by the algebraically smooth vectors of the graph Laplacian, associated with an adaptive algebraic multigrid (AMG) method for solving linear systems. More specifically, we generate a sequence of m vectors $\mathbf{q}_1, \dots, \mathbf{q}_m$, where each new vector $\mathbf{q}_k$ is computed on the basis of the previously computed ones $\mathbf{q}_j, j = 1, \dots, k-1$. These $k-1$ vectors are incorporated in $k-1$ AMG hierarchies and define a composite AMG solver $B := B_{k-1}$ composed of $k-1$ V-cycles. The solver B is applied in a stationary iteration method to solve the trivial system $L_S\mathbf{x} = \mathbf{0}$ starting with a random initial guess. By monitoring the error (note that the exact solution is $\mathbf{x} = \mathbf{0}$, so access to the error is available), we can very accurately measure the convergence rate of that composite AMG method. If the method has not reached a prescribed quality (in terms of desired convergence factor), the iteration process exposes an *algebraically smooth* vector in the lower portion of the spectrum of $B^{-1}L_S$, namely, the most recent iterate. That iterate (after normalization) gives rise to the new vector $\mathbf{q}_k$. In the next step k, we build one more AMG hierarchy using $\mathbf{q}_k$ to guide the AMG process and define the new composite solver $B := B_k$, and repeat. The process continues until the method reaches a desired prescribed convergence factor. It is possible to show that the space spanned by the above vectors gives an accurate approximation to the lower portion of the spectrum of a suitable generalized eigenvalue problem associated with L_S. Using this space for the graph embedding, a standard *K-means* algorithm[6] is applied for data clustering. This classifies our method as a spectral clustering one.

We evaluate the accuracy of the proposed methodology by comparing the estimated clustering with the outcome of some popular clustering algorithms available in the **R** software framework.[7] The comparison is performed by measuring modularity function obtained by the different algorithms, as well as standard clustering similarity measures,[8,9] such as Variation of Information (VI). Experiments are carried out on a large set of similarity graphs coming both from commonly used benchmarks and from real-life data sets.

This paper is organized as follows. In Section 2, we provide a brief background on spectral clustering exploiting the graph Laplacian and describe some basic algorithms and quality metrics for clustering. In Section 3, we introduce the concept of algebraically smooth vectors for graph Laplacian and discuss the rationale of our method for computing an efficient low-dimensional graph representation as well as its relation with standard spectral clustering techniques. In Section 4, we describe some main features of the data sets used for demonstrating the feasibility of our method as a data clustering tool, while Section 5 includes some discussion on the obtained results. Concluding remarks and future work are summarized in Section 6.

2 | BACKGROUND

2.1 | Laplacian eigenvectors and graph partitioning

Given a partition $V_1, \dots, V_K$, let $\mathbf{h}_k = (h_{1k}, \dots, h_{nk})^T$ be the K vectors that map the graph vertices to the partition sets, defined as

$$h_{ik} = \begin{cases} 1/\sqrt{|V_k|}, & \text{if } x_i \in V_k \\ 0, & \text{otherwise} \end{cases} \quad i = 1, \dots, n; \; k = 1, \dots, K. \tag{2}$$

Let $H \in \mathcal{R}^{n \times K}$ be the matrix whose columns are the K mapping vectors; H has orthonormal columns, that is, $H^T H = I$, where I is the identity matrix of dimension K. It is simple to show the following equalities[2]:

$$\mathbf{h}_k^T L \mathbf{h}_k = \frac{1}{2} \frac{W(V_k, \overline{V_k})}{|V_k|} = (H^T L H)_{kk},$$

then, it holds

$$RatioCut(V_1, \dots, V_K) = \sum_{k=1}^{K} \mathbf{h}_k^T L \mathbf{h}_k = \sum_{k=1}^{K} (H^T L H)_{kk} = \mathrm{Tr}(H^T L H),$$

where $\mathrm{Tr}(\cdot)$ is the trace of a matrix. Therefore, the minimization of (1) can be reformulated in terms of the following trace minimization problem:

$$\min_{V_1, \dots, V_K} \mathrm{Tr}(H^T L H), \quad \text{subject to } H^T H = I, \text{ with columns of } H \text{ defined as in (2).}$$

In practice, we relax the constraint (2) for the vectors $\mathbf{h}_k$, allowing them to take arbitrary real values, the solution of this relaxed minimization problem is obtained by choosing for $\mathbf{h}_k$ the first K eigenvectors of L. These K eigenvectors are used as coordinate vectors for the graph vertices, which allows to apply standard spatial clustering algorithms, such as the well-known K-means. It is known that, in the case of well-clustered graphs, ie, where communities are well separated, the above spectral representation of the graph works quite well, giving a good approximation of optimal partitioning.[10] We note that the above formulation penalizes clustering in which either of the sets is small and favors balanced divisions over unbalanced ones; therefore, objective functionals alternative to (1), which correspond to normalized versions of Laplacian, are often used in practice; see the works of von Luxburg[2] and Newman[11] for a discussion. We point out here that our method can similarly be applied to each such symmetric positive definite (s.p.d.) versions of modified Laplacian matrix.

2.2 | K-means algorithm

After embedding (ie, assigning coordinates to each vertex of the graph), the problem of clustering reduces to data partition in a Euclidean space. This is one of the oldest and most important task in computational geometry; it can be formulated in terms of the K-means problem: given an integer K and a set of n data points in R^{n_c} ($n_c \ll n$), the objective is to choose K centers that minimize the total squared distance between each point and its closest center. The K-means algorithm is a local search optimization method.[6] It seeks to find a partition $V_1, \dots, V_K$ of K circular sets with centers $\mathbf{c}_1, \dots, \mathbf{c}_K$, which minimizes the sum of the squared Euclidean distance between $\mathbf{x}_i \in R^{n_c}$ and the center of the set to which it is assigned

$$\sum_{k=1}^{K} \sum_{\mathbf{x}_i \in V_k} \|\mathbf{x}_i - \mathbf{c}_k\|^2.$$

The K-means algorithm, described in Algorithm 1, starts with K arbitrary centers, typically chosen uniformly at random from the data points. Each point is then assigned to the nearest center, and each center is recomputed as the center of mass of all points assigned to it: $(\mathbf{c}_k)_j = 1/|V_k| \sum_{\mathbf{x}_i \in V_k} (\mathbf{x}_i)_j \; \forall j = 1, \dots, n_c$. This version of the algorithm provides a simple and fast method for spatial clustering, although it offers no approximation guarantees; indeed, the final result largely depends on the initial centers and it can lead to a poor approximation of the global minimum of the objective function; however, the method is still among the widely used ones in practice after applying spectral embedding of the graphs.[2]

Algorithm 1 *K-means*

```
Data: A set of points $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ with $\mathbf{x}_i \in \mathcal{R}^{n_c}$, an integer $K < n$, an integer $maxiter$
Result: A partition $\{V_k\}$ of $X$ into $K$ nonempty clusters
Initialize $K$ centers $\mathbf{c}_k$ by sampling $K$ random points from $X$
while $\mathbf{c}_k$ is changing and $iter < maxiter$ do
    $V_k = \emptyset$
    for $\mathbf{x}_i \in X$ do
        find $k = \arg\min_{1 \le k \le K} \|\mathbf{x}_i - \mathbf{c}_k\|^2$
        add $\mathbf{x}_i$ to $V_k$
    end for
    compute new centers: $(\mathbf{c}_k)_j = 1/|V_k| \sum_{\mathbf{x}_i \in V_k} (\mathbf{x}_i)_j \; \forall j = 1, \dots, n_c$
end while
```

2.3 | Quality metrics

In this work, we focus on the feasibility of our method in obtaining good quality clusterings. In network science, it is important to be able to analyze the quality of clustering algorithms and compare different methods. To accomplish this, we rely on some standard quality metrics and refer the reader to the work of Carissimo et al[9] for specific discussion and methodologies for sensitivity analysis.

A well-accepted popular measure of the quality of graph partitioning is the modularity functional.[11] It is defined as the fraction of the edges that fall within the groups minus the expected value of such fraction if edges were distributed at random. Let A be the adjacency matrix of our graph G. Given a partition of the graph and denoted by V_i, the set from the partition to which vertex i is assigned, the number of edges that fall within the sets, for this particular partition, is equal to $1/2\sum_{ij} A_{ij}\delta_{V_iV_j}$, where $\delta_{V_iV_j}$ is the Kronecker symbol. Suppose that, keeping the total number of edges the same as the original graph and also preserving the degree of every vertex, we reposition the edges at random. Let $P_{ij} = \frac{k_ik_j}{2m}$ be the probability that vertices i and j are connected by an edge, where $k_i = \sum_j A_{ij}$ is the degree of vertex i and m is the total number of edges, then the expected number of edges within same sets, post-randomization, equals $1/2\sum_{ij} \frac{k_ik_j}{2m}\delta_{V_iV_j}$. With the above common choice of randomization, the modularity functional reads

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_ik_j}{2m} \right) \delta_{V_iV_j}. \tag{3}$$

We observe that $Q \in [-1, 1]$. By definition of Q, graphs with high modularity have a dense set of connections between vertices within the same set or *module* but sparser connections between vertices across different modules. Thus, modularity is the state-of-the-art tool employed in detecting community structures within large graphs and graphs with a strong community structure have high modularity. Indeed, many clustering algorithms are based on algorithms that maximize modularity (see, for example, the works of Clauset et al[12] and Blondel et al[13]).

In this paper, we also employ another metric, referred to as VI,[8,9] used to measure the quality of a partition and to compare partitions. Consider graph G and two partitions $C = (C_k)_{k=1}^{K}$ and $C' = (C_{k'})_{k'=1}^{K'}$, with K and K' nonempty sets, respectively. We define $P(k) = |C_k|/n$ to be the probability of a vertex being in the set C_k, $P(k') = |C_{k'}|/n$ the probability of a vertex being in the set $C_{k'}$, and $P(k, k') = |C_k \cap C_{k'}|/n$ as the probability that a point belongs to the set C_k in partition C and to set $C_{k'}$ in partition C'. Then, VI is defined as

$$VI(C, C') = H(C) + H(C') - 2I(C, C'), \tag{4}$$

where $H(C)$ is the entropy associated with partition C

$$H(C) = -\sum_{k=1}^{K} P(k) \log(P(k)),$$

$H(C')$ is the entropy associated with partition C', and $I(C, C')$ is the mutual information between C and C', ie, the information that one partition has about the other

$$I(C, C') = \sum_{k=1}^{K} \sum_{k'=1}^{K'} P(k, k') \log \frac{P(k, k')}{P(k)P(k')}.$$

We have that $VI \in [0, \log(n)]$ and it represents a distance measure between the two different partitions.

3 | BOOTSTRAP AMG FOR GRAPH EMBEDDING

The spectral methods for graph embedding generally use the first $n_c \geq 1$ eigenvectors corresponding to the lower portion of the spectrum of the graph Laplacian associated with the given graph: $L\mathbf{q}_k = \lambda_k \mathbf{q}_k$, where $\lambda_1 \leq \lambda_2 \leq \cdots \lambda_n$.

In this work, we propose to use as low-dimensional space for graph embedding, the space spanned by the algebraically smooth vectors generated by a composite iterative process, where at each step, the quality of the currently available solver is tested. If that solver has not reached a pre-selected convergence factor, the composite solver is updated with one more component, and the process is repeated. To build the new component of the solver an algebraically smooth vector is utilized, which guides the construction of a specific AMG solver of V-cycle type (to maintain linear complexity). For our given matrix L_S and a current solver B, an algebraically smooth vector is constructed as follows. We solve the trivial equation $L_s\mathbf{x} = \mathbf{0}$ by iterations starting with a random $\mathbf{x}_0$ and consecutively compute

$$\mathbf{x}_\ell = (I - B^{-1}L_S)\mathbf{x}_{\ell-1}, \ \ell = 1, \ldots, \ell_{\max}.$$

Note that the iterates $\mathbf{x}_\ell$ are actually the true errors (since the exact solution is the null vector). This allows us to estimate the convergence factor ϱ of the iteration process very accurately. If the estimated factor stays above a pre-selected desired factor ϱ_{desired} after $\ell_{\max}$ iterations, the current iterate $\mathbf{x}_\ell$ is declared to be *algebraically smooth* in the sense that it is spanned mostly by the eigenvectors in the lower portion of the spectrum of $B^{-1}L_S$. After m such steps, we end up with a composite solver B_{BAMG} that has the representation $I - B_{\text{BAMG}}^{-1}L_S = \prod_{r=1}^{m}(I - B_r^{-1}L_S)$, where each component B_r is a V-cycle AMG constructed on the basis of the rth smooth vector $\mathbf{q}_r$ (which is the corresponding iterate $\mathbf{x}_{\ell_{\max}}$ at the rth step of the bootstrap process, after normalization). The application of the composite solver B_{BAMG}, which is a composition of coarse-grid solves complemented by smoothing iterations over hierarchy of levels, effectively removes all error components whereas its coarsest level solves, by construction, eliminate all error components associated with the space spanned by the set of algebraically smooth vectors $\{\mathbf{q}_r\}_{r=1}^m$. In summary, the bootstrap AMG solver exploits implicitly two processes, the elimination of the error components associated with the space spanned by the vectors $\mathbf{q}_1, \ldots, \mathbf{q}_m$ and a complementary process, composed by smoothing iterations over hierarchy of levels, giving rise to a mapping M. In the next subsection, we describe the relation of the convergence of the bootstrap AMG solver with a specific generalized eigenvalue problem that serves as a motivation of using the set $\{\mathbf{q}_r\}_{r=1}^m$ for graph embedding similar to standard spectral clustering methods.

3.1 | Rationale to use algebraically smooth vectors

In this section, we describe the construction of a set of algebraically smooth vectors of the graph Laplacian L_S to serve as coordinate directions in embedding a given graph into $\mathcal{R}^{n_c}$, with $1 \leq n_c \ll n$. We show that the estimated smooth vectors span a coarse space that represents well the global information associated with the original graph relying on the convergence theory of two-level hierarchical methods.[14]

Given the modified graph Laplacian matrix L_S, we can define a two-level method for solving the linear system $L_S\mathbf{x} = \mathbf{b}$. Let V_c be the space spanned by a set of smooth vectors $\{\mathbf{q}_r\}_{r=1}^m$ of L_S with respect to a s.p.d. operator M, having the composite form $M = M_1^T(M_1 + M_1^T - L_S)^{-1}M_1$, with M_1 being invertible. Let $\{\phi_i\}_{i=1}^{n_c}$ provide a basis of V_c, and let $P = [\phi_1, \ldots, \phi_{n_c}]$ be the interpolation matrix mapping vectors from V_c to V and $(L_S)_c = P^TL_SP$ the corresponding coarse Laplacian matrix. Consider the following two-level iterative method.

- Initialize $\mathbf{x} := \mathbf{0}, \mathbf{r} := \mathbf{b}$
- *Pre-smoothing:* Solve $M_1\mathbf{y} = \mathbf{r}$ and update $\mathbf{x} := \mathbf{x} + \mathbf{y}, \mathbf{r} := \mathbf{b} - L_S\mathbf{x}$
- Solve the coarse problem: $(L_S)_c\mathbf{x}_c = \mathbf{r}_c := P^T\mathbf{r}$
- Update the iterate: $\mathbf{x} := \mathbf{x} + P\mathbf{x}_c$ and the respective residual $\mathbf{r} := \mathbf{b} - L_S\mathbf{x}$
- *Post-smoothing:* Solve $M_1^T\mathbf{z} = \mathbf{r}$
- Update $\mathbf{x} := \mathbf{x} + \mathbf{z}$

The iteration matrix of the above two-level iteration process can be written in the form

$$B^{-1} = M^{-1} + \left(I - M_1^{-T}L_S\right)P(L_S)_c^{-1}P^T\left(I - L_SM_1^{-1}\right),$$

where $M = M_1(M_1 + M_1^T - L_S)^{-1}M_1^T$ is the so-called symmetrized smoother, such that $I - M^{-1}L_S = (I - M_1^{-T}L_S)(I - M_1^{-1}L_S)$. Assuming that M_1 is a symmetric and convergent smoother in the L_S-norm, the following spectral equivalence result holds[14]:

$$\mathbf{v}^T L_S \mathbf{v} \le \mathbf{v}^T B \mathbf{v} \le C\, \mathbf{v}^T L_S \mathbf{v},$$

where the optimal constant C is given by the formula

$$C = \max_{\mathbf{v}} \frac{\|\mathbf{v} - \pi \mathbf{v}\|_M^2}{\|\mathbf{v}\|_{L_S}^2}. \tag{5}$$

Here, $\pi : V \mapsto V_c \subset V$ is the projection in the M-inner product. The spectral equivalence result shows that, if we can keep the iterates M-orthogonal to the coarse space, the error will be efficiently reduced by the composite smoother M. In other words, by the nature of our bootstrap AMG solver (defined in the previous subsection), the *essential global* information is propagated by the coarse solve due to the coarse space V_c spanned by the *algebraically smooth* vectors $\{\mathbf{q}_r\}_{r=1}^m$, whereas the complementary process corresponding to the smoothing steps with M_1 and M_1^T will have more local nature (with respect to the hierarchy of levels). The following result will make the above statement more precise.

Theorem 1. *Consider the operator B defined by the two-level algorithm with smoother M_1, symmetrized smoother M, and coarse space V_c. Consider the generalized eigenvalue problem*

$$L_S \mathbf{q}_k = \lambda_k M \mathbf{q}_k, \tag{6}$$

where $0 \le \lambda_1 \le \lambda_2 \le \cdots \le \lambda_m < \lambda_{m+1} \le \cdots \le \lambda_{\max}$, and assume that the coarse space V_c contains the first $m \ge 1$ eigenvectors $\mathbf{q}_k$. Then, the following estimate holds for the spectral equivalence constant C (or the corresponding convergence factor ϱ):

$$C = \frac{1}{1-\varrho} \le \frac{1}{\lambda_{m+1}}.$$

Proof. Using the eigenvectors $\{\mathbf{q}_k\}$ of (6), which are chosen to be M-orthonormal, we can decompose any vector $\mathbf{v}$ as $\mathbf{v} = \sum_k v_k \mathbf{q}_k$. Since the coarse space contains the first m eigenvectors $\mathbf{q}_k$, for the M-orthogonal projection π onto the coarse space V_c, we have that

$$\begin{aligned}
\|\mathbf{v} - \pi\mathbf{v}\|_M^2 &= \min_{\mathbf{v}_c \in V_c} \|\mathbf{v} - \mathbf{v}_c\|_M^2 \\
&\le \min_{\{w_k\}_{k=1}^m} \left\| \mathbf{v} - \sum_{k=1}^m w_k \mathbf{q}_k \right\|_M^2 \\
&= \sum_{k>m} v_k^2 \\
&\le \frac{1}{\lambda_{m+1}} \|\mathbf{v}\|_{L_S}^2 .
\end{aligned}$$

The latter shows the desired estimate $C = \frac{1}{1-\varrho} \le \frac{1}{\lambda_{m+1}}$. □

In the bootstrap AMG algorithm, described in the previous subsection, we compute a set of vectors $\{\mathbf{q}_r\}_{r=1}^m$ that effectively try to ensure that the constant C in (5) is below a certain value $\frac{1}{1-\varrho_{\text{desired}}}$. That is, we do not actually solve the above generalized eigenvalue problem (6); instead, we select vectors $\mathbf{q}_k$, one at the time, by monitoring the error that the currently constructed AMG solver is unable to reduce effectively. These vectors are used to build an improved hierarchy until we obtain a solver with desired convergence factor $\varrho = \varrho_{\text{desired}}$. In other words, we have by construction

$$C \le \frac{1}{1-\varrho} \simeq \frac{1}{\lambda_{m+1}}.$$

This indicates that our bootstrap AMG constructed vectors $\{\mathbf{q}_r\}_{r=1}^m$ approximate very well the space spanned by the lower portion of the spectrum of $M^{-1}L_S$; therefore, in that sense, it is related to the more classical spectral clustering, although it effectively uses a different spectral problem. We also note that we do not actually have M explicitly, rather we have the inverse actions of $B = B_{\text{BAMG}}$ (which implicitly define M^{-1} as a complementary process to the coarse solves).

We mention that, using the notion of algebraically smooth vectors for defining distance on graphs has been studied previously, see, for example, the works of Ron et al[15] and Chen and Safro.[16] Our approach can be viewed as an extension of the above approaches, since we utilize a more general B in a multilevel setting, whereas previously B was chosen to be a single-level smoother, as the Jacobi method, which cannot ensure estimates of the form (5) for a small number of vectors, while bootstrap AMG makes it feasible to obtain small ϱ. Furthermore, we note that an efficient AMG method specifically tailored for fast graph Laplacian linear solver was introduced in the work of Livne and Brandt,[17] while other adaptive/bootstrap AMG methods, meant as reliable and flexible linear solvers when no a priori information on the near-kernel space of the linear operator is available, were previously introduced in the works of Brezina et al[18,19] and Brandt et al.[20]

3.2 | Some details on using bootstrap AMG

In our bootstrap process, we build a linear solver B composed from a number of different AMG cycles leading to the following error propagation matrix:

$$I - B^{-1}L_S = \left(I - B_r^{-1}L_S\right)\cdots\left(I - B_1^{-1}L_S\right)\left(I - B_0^{-1}L_S\right), \tag{7}$$

where each B_r is an AMG cycle operator built by a suitable coarsening algorithm, referred to as *coarsening based on compatible weighted matching*. For details on the coarsening scheme and the bootstrap AMG, we refer the reader to other works.[21,22] Each B_r is an AMG V-cycle built on the basis of a conducted by the method (algebraically smooth) vector $\mathbf{q}_r$.

The process is iterated and m vectors $\{\mathbf{q}_r\}_{r=1}^m$ can be computed. As discussed previously, these vectors represent approximations of the eigenvectors of preconditioned versions of the graph Laplacian L_S. The number m of computed vectors, in general, depends on the desired factor ϱ we use to build the composite solver B_{BAMG} defined from the product iteration matrix

$$I - B_{\text{BAMG}}^{-1}L_S = \left(I - B_m^{-1}L_S\right)\cdots\left(I - B_1^{-1}L_S\right)\left(I - B_0^{-1}L_S\right).$$

For a given $\varrho \in (0, 1)$, the number of components m (equivalently, the number of smooth vectors associated with the components B_r) is such that by construction, we have

$$\|I - B_{\text{BAMG}}^{-1}L_S\|_{L_S} \leq \varrho.$$

In order to obtain a set of orthogonal vectors, we apply a singular-value decomposition (SVD) to the computed set of vectors $\{\mathbf{q}_r\}_{r=1}^m$ and use the ith component of each left-singular vector $\{\mathbf{u}_r\}_{r=1}^{n_c}$ as coordinate vector of the general point x_i in the vector space spanned by the smooth vectors computed with our bootstrap AMG. More precisely, we add to the n_c orthogonal vectors obtained by SVD of the smooth vectors also the vector of all ones $(1, \dots, 1)^T$, as representative eigenvector corresponding to the null eigenvalue of L, also used as starting smooth vector in the bootstrap process. This new representation of the original data set is employed as input of a *K-means* algorithm to generate clusters of the original data set. Our algorithm for spectral clustering is described in Algorithm 2.

Algorithm 2 *Bootstrap AMG for spectral clustering*

```
Data: $L_S$: matrix, $\mathbf{q}_0$: unit vector, $\varrho_{\text{des}}$: desired conv. rate, $maxiter$: max num of bootstrap iterations, $\ell_{\max}$: num of iterations for testing phase
Result: Clusters $V_1, \dots, V_K$ with $V_i = \{j \mid y_j \in V_i\}$
Initialize: $r = 1$, $\varrho = 1.0$
while $\rho \geq \rho_{\text{des}}$ and $r \leq maxiter$ do
    Building Phase: build a new AMG operator $B_r$
    Testing Phase: apply composite AMG and compute new smooth vector:
        let $\mathbf{q}_r^0$ a random vector
        apply $\mathbf{q}_r^\ell = \prod_r (I - B_r^{-1}L_S)\mathbf{q}_r^{\ell-1}$,  $\ell = 1, \dots, \ell_{\max}$
        estimate $\varrho$
        $\mathbf{q}_r = \mathbf{q}_r^{\ell_{\max}} / \|\mathbf{q}_r^{\ell_{\max}}\|$
end while
orthogonalize $\{\mathbf{q}_r\}$ by SVD computation
let $U \in \mathcal{R}^{n \times (n_c+1)}$ be the matrix containing the left orthogonal vectors and the constant (unit) vector as columns
apply K-means to $y_i \in \mathcal{R}^{n_c+1}$ corresponding to the $i$th row of $U$
```

We implemented Algorithm 2 in the context of the BootCMatch software framework, which is an open-source C code implementing all the functionalities for building and applying the boostrap AMG involved in this work and described in other works.[21,22]

4 | BENCHMAK DATA SETS

In the following, we discuss results of experiments carried out both on synthetic graphs obtained by a well-known generative model and on some freely-available real-life networks. Our main aim here is to evaluate the ability of our procedure to define a suitable low-dimensional representation of the graphs, which preserves its key information and allows us to identify possible group structures using standard K-means. In what follows, we describe the main features of the test graphs.

4.1 | Synthetic graphs generated by stochastic block model

The Stochastic Block Model (SBM) is a generative model for random graphs, which assigns a probability value to each edge (i,j) of the graph of dimension n. In the basic model, the SBM is defined by a scalar value K, defining the number of groups in the network; a vector $\mathbf{g}$ of dimension n, where g_i gives the group index of vertex i; a $K \times K$ matrix M, where the matrix entry $M_{l_1 l_2}$ is the probability that a vertex in group l_1 is connected to a vertex in group l_2. Given the above parameters, to each pair i,j is assigned a probability of forming an edge by first looking at $\mathbf{g}$, which gives g_i and g_j, and then looking at the entry M_{g_i,g_j} of M to find the probability that such an edge exists.

In our simulations, we generated random network graphs using the Degree Corrected SBM (DCSBM). Similarly to the examples provided in the work of Carissimo et al[9] and Cutillo and Signorelli,[23] we implemented the approach proposed in the work of Karrer and Newman.[24] The DCSBM model is a modification of SBM in order to obtain more realistic graphs. Indeed, SBM assumes that vertices within a block have the same degree that in contrast to many real-life networks; whereas in the DCSBM, the probability value for an edge between nodes depends both on the communities to which the vertices belong and on specified vertex weights. These weights are defined in a way such that the average vertex weight in each block is equal to 1 and allows us to set variable vertex degrees in the communities. In particular, we simulated different scenarios where, per each graph, we fixed a unique edge probability, M_{out}, between any couple of communities, while the edge probability within each community, M_{in}, was uniformly generated in $[0.3, 0.7]$, leading to an average value $M_{\text{in}} = 0.5$. Varying the value of $M_{\text{out}} \in [0.001, 0.8]$ allows us to directly control the modularity; indeed, a low value of M_{out} corresponds to a high value of modularity and vice-versa. We considered a total of 144 graphs of increasing dimension $n = 1000, 2000, 3000, 4000$ and sparsity degree ranging in $[0.01, 0.35]$. For each dimension, we considered different values for $K = 4, 8, 12, 16$ and, varying M_{in} and M_{out} as explained before, we selected nine graphs with increasing modularity for each K, leading to a total of 36 graphs, numbered from 1 to 36 corresponding to increasing values of K.

4.2 | Real-life networks

We also discuss some results of experiments obtained on realistic networks where we have no information about possible community structure. We considered a small size biological network from the data set available in the igraphdata package for the **R** software framework,[7] and some increasing in size information and technological networks from the work of Newman[25] and the DIMACS 10th challenge collection,[26] which includes various graphs particularly challenging for graph partitioning and clustering. The main features of the selected graphs are summarized in Table 1 where we report the name of the graph (*Name*), the number of graph vertices (n), the number of its edges (nnz), the maximum and minimum vertex degree (*maxdeg* and *mindeg*), and the sparsity degree ($2nnz/(n^2 - n)$). Note that, in the cases of original graphs with more than one connected component, we chose the connected component with the largest number of vertices, which we obtained using a C implementation of the Tarjan algorithm.[27]

In Figure 1, we visualize the smallest graph and the sparsity pattern of its adjacency matrix. It represents the immunoglobulin interaction network whose vertices correspond to amino acids and an edge is drawn between two amino acids if the shortest distance between their Cα atoms is smaller than the threshold value $\theta = 8$ Ångström.

TABLE 1 Main features of selected graphs (maximum connected component)

Name	*n*	*nnz*	*maxdeg*	*mindeg*	*sparsity*
immuno	1316	6300	17	3	7.28×10^{-3}
as22july-06	22963	48436	2390	1	1.84×10^{-4}
email-Enron	33696	180811	1383	1	3.19×10^{-4}
ct2010	67578	168176	53	1	7.37×10^{-5}
caidaRouterLevel	190914	607610	1071	1	3.33×10^{-5}

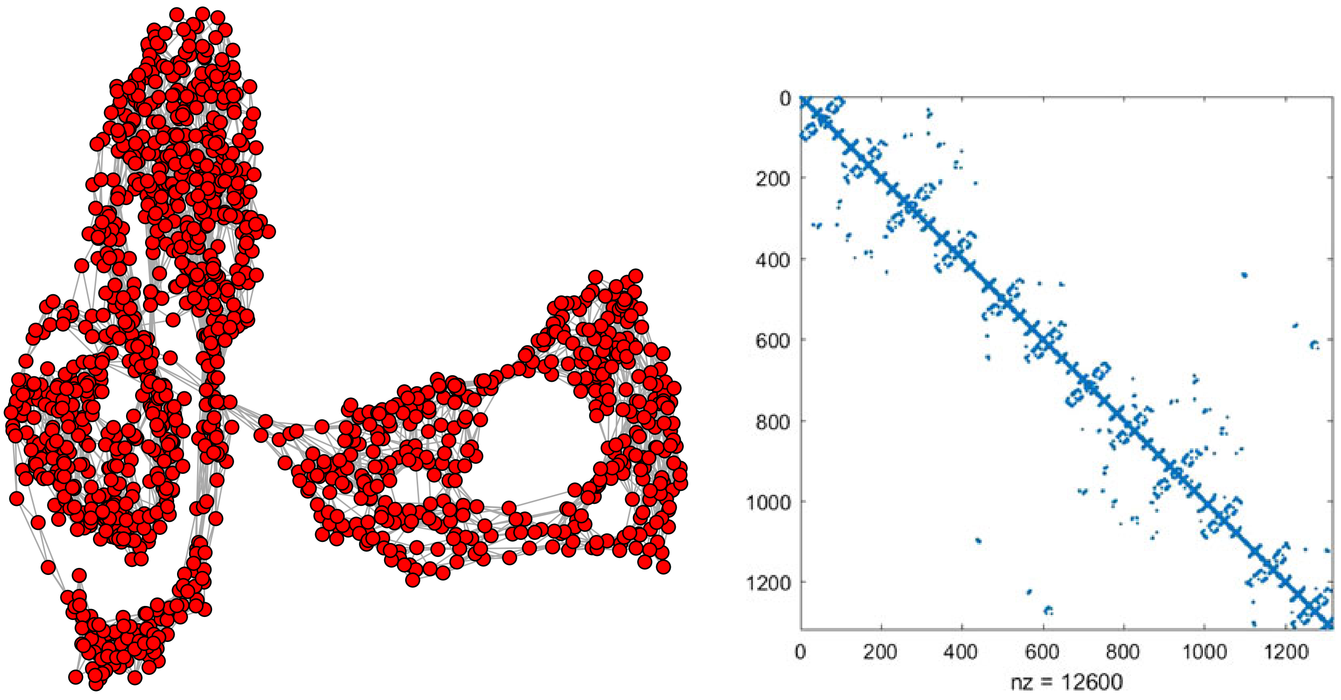

FIGURE 1 Immunoglobulin interaction network. Graph (left) and sparsity pattern of adjacency matrix (right)

5 | RESULTS

We discuss results of the application of Algorithm 2 to the selected graphs. Our procedure is implemented in BootCMatch, but for the final spatial clustering, at the present, we rely on the K-means MATLAB implementation. In order to reduce sensitivity of the K-means algorithm with respect to the starting cluster centers, we run Algorithm 1 for 100 times with the same input data and chose the partition corresponding to the maximum value of the computed modularity. In the application of our method, we built m smooth vectors, where $m = \min(40, \overline{K})$, with $\overline{K}$ the number of AMG operators built to obtain a final multiplicative operator of type (7) with a desired convergence rate less than $\varrho = 10^{-8}$. Each AMG operator is a symmetric V-cycle with one forward/backward Gauss-Seidel sweep as pre/post-smoothing. A direct LU factorization is used as coarsest solver. Default parameters are used for the coarsening implemented in BootCMatch, for details, we refer the reader to other work.[22] In these experiments, we used $\ell_{\max} = 15$ iterations to estimate both a new smooth vector and the convergence rate of the composite bootstrap AMG of type (7) at each bootstrap iteration. The experiments were carried out with BootCMatch rel. 0.9 on one core of a 2.6 GHz Intel Xeon E5-2670, running Linux 2.6 and GNU compiler 4.6.

5.1 | Results on DCSBM graphs

In Figure 2, we show a clustering obtained by our bootstrap method for one of the selected graphs. In detail, on the left, we have the original graph with $n = 1000$ vertices, corresponding to $K = 4$ blocks and a true modularity $Q = 0.62$. On the right, we report the picture obtained by projecting the graph vertices in the four-dimensional Euclidean space generated by three smooth vectors built by our bootstrap procedure and the unit vector. Different colors represent the

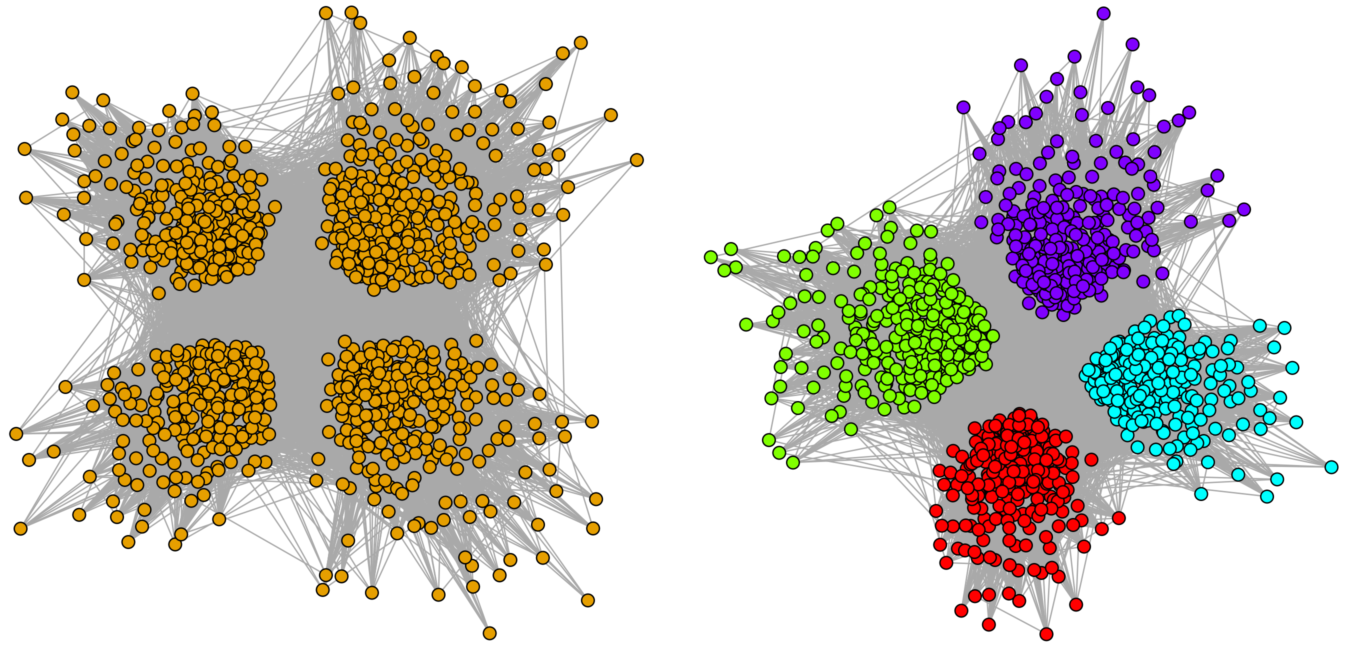

FIGURE 2 Original graph (left); clustering obtained with BootCMatch (right)

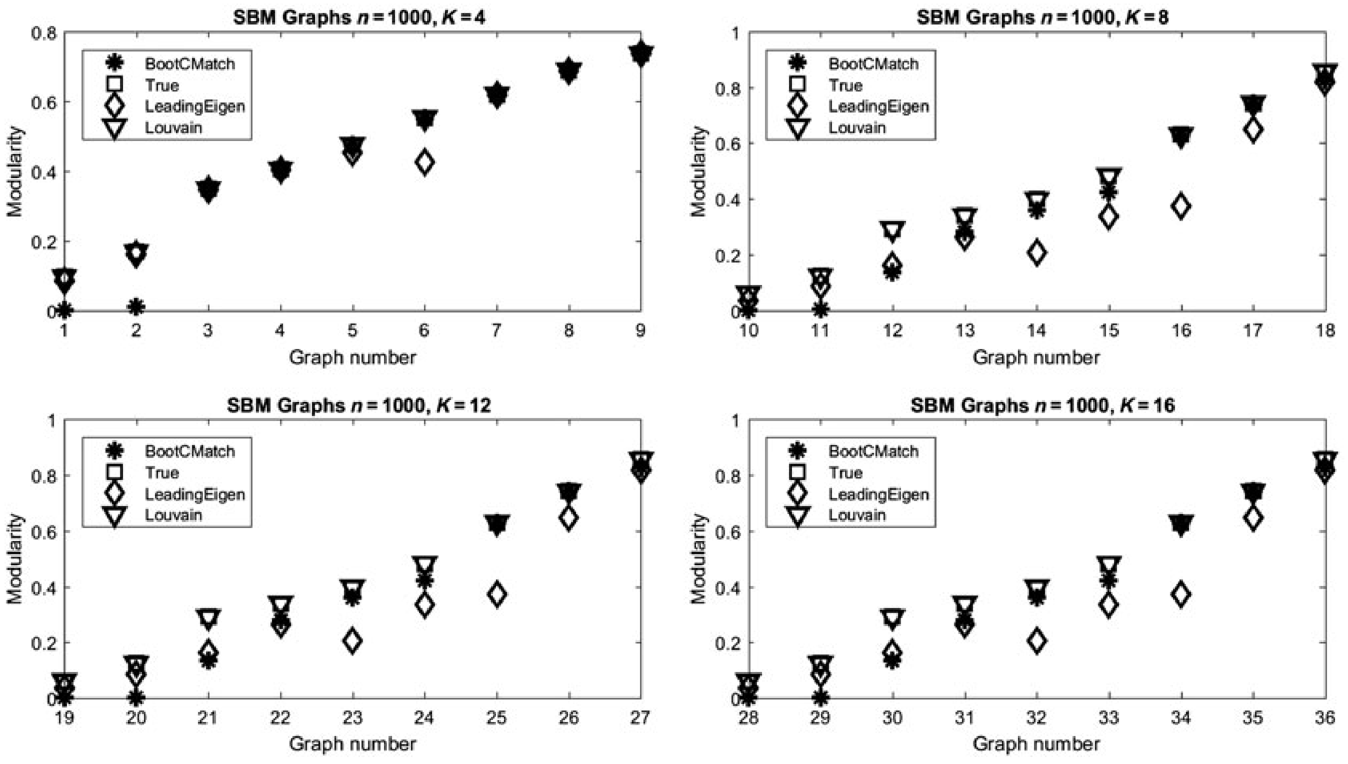

FIGURE 3 Stochastic Block Model (SBM) graphs ($n = 1000$): comparison of modularity values among different clustering

different groups identified by the K-means algorithm. We can see that our method is able to detect four well-separated groups with high connection density in the network and compute a clustering corresponding to a modularity of 0.6192.

In Figures 3 to 6, we report modularity values, for different values of K and for all the considered graph dimensions n. We compare modularity values of clustering computed for data sets represented by the vectors obtained by BootCMatch with expected true modularities. For comparison aim, we also considered modularities obtained by applying two algorithms available in the **R** software environment. The *cluster_leading_eigen* function (LeadingEigen), which implements

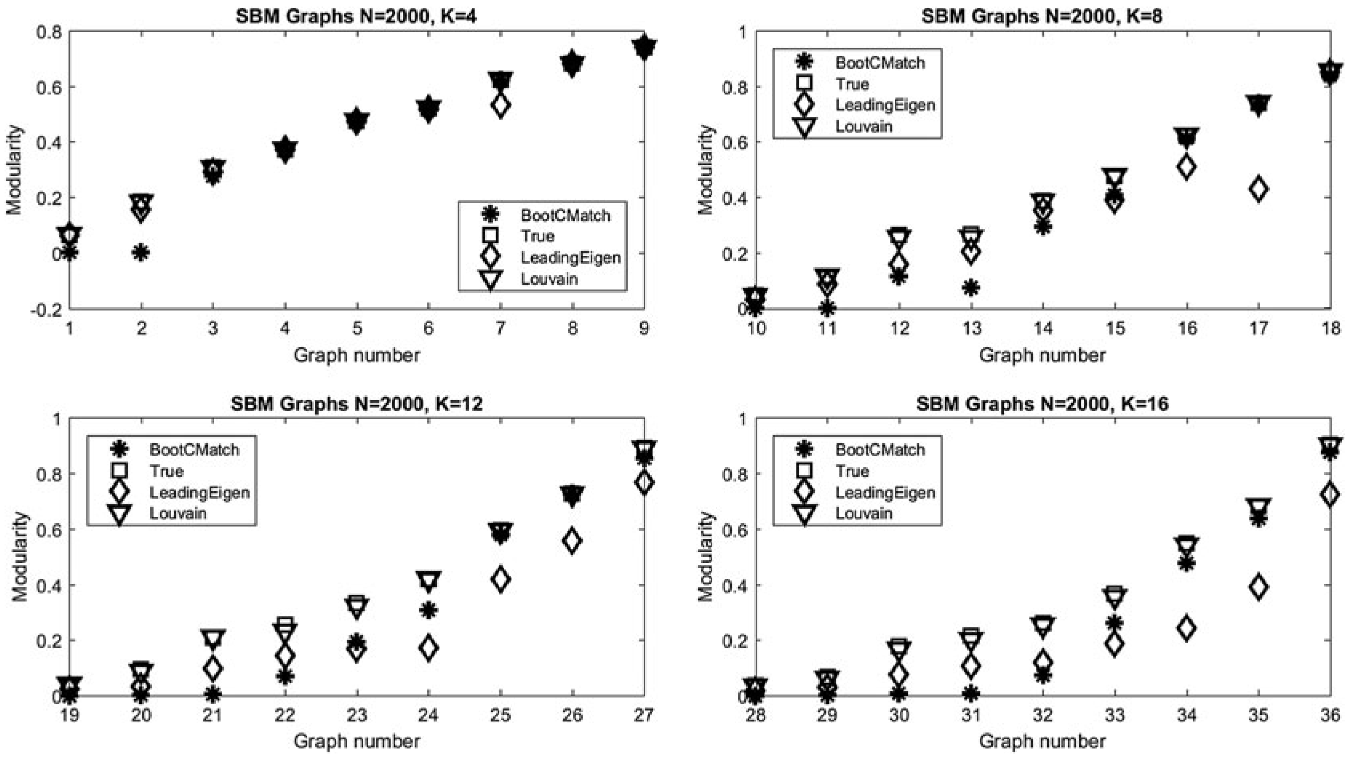

FIGURE 4 Stochastic Block Model (SBM) graphs ($n = 2000$): Comparison of modularity values among different clustering

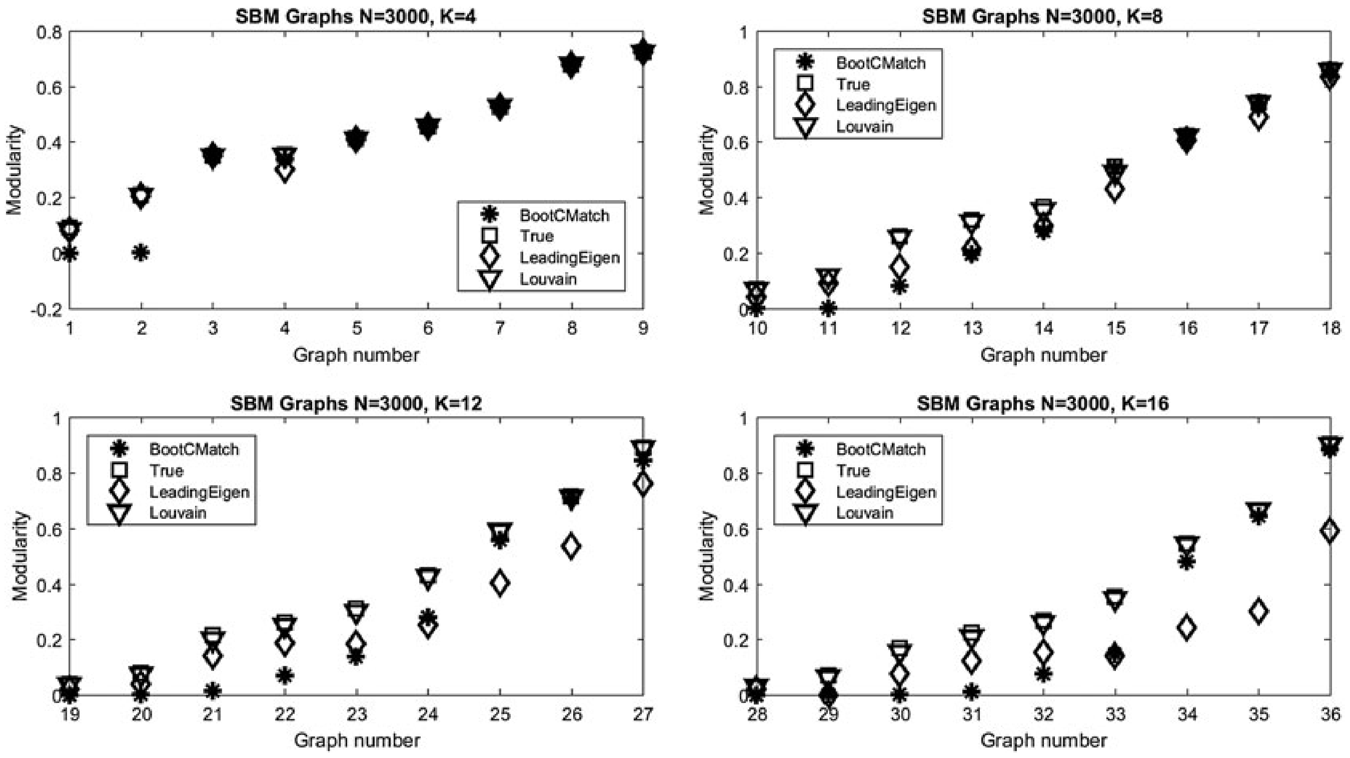

FIGURE 5 Stochastic Block Model (SBM) graphs ($n = 3000$): Comparison of modularity values among different clustering

the *leading eigenvector* method described in the work of Newman,[28] based on the computation of the eigenvector corresponding to the largest positive eigenvalue of the modularity matrix associated to the modularity functional defined in (3) and the *cluster_louvain* function (Louvain), implementing the greedy modularity optimization method described in the work of Blondel et al.[13]

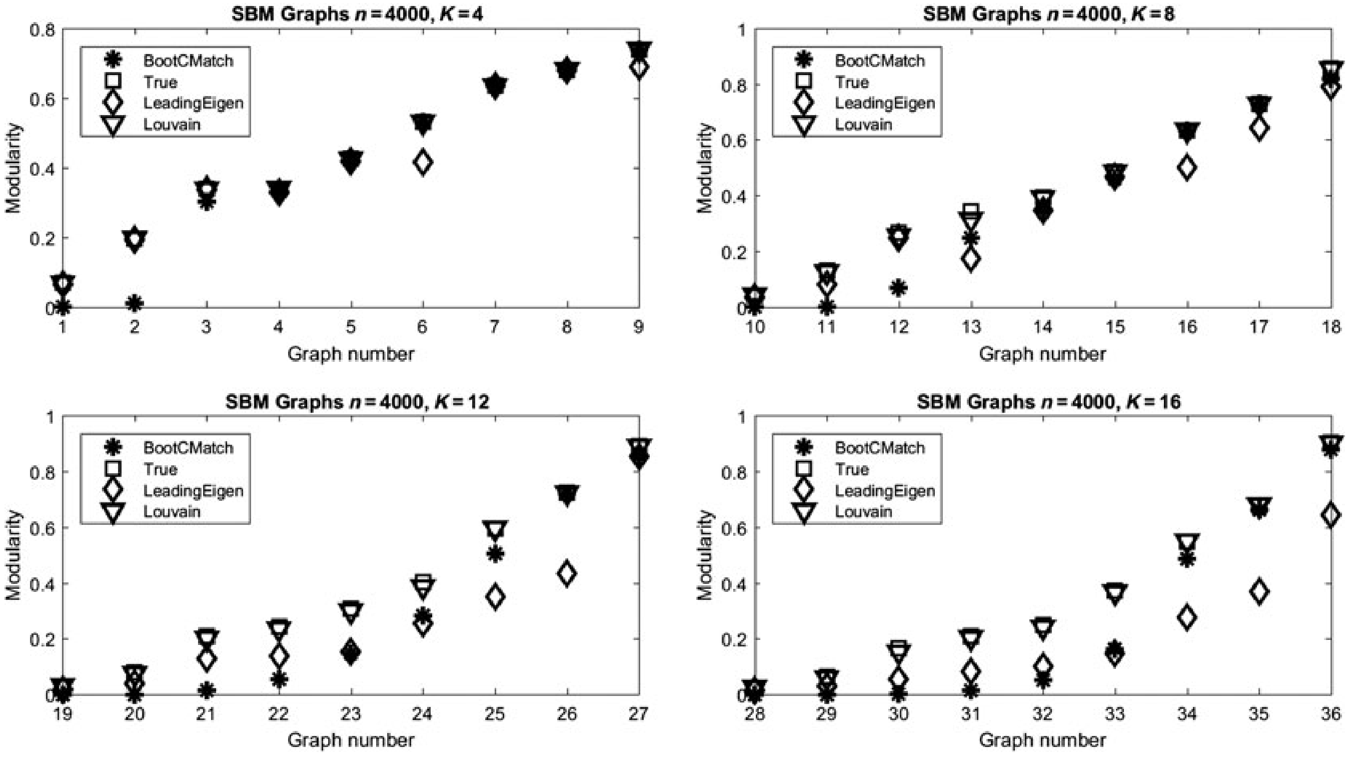

FIGURE 6 Stochastic Block Model (SBM) graphs ($n = 4000$): comparison of modularity values among different clustering

We observe that our method obtains good values for modularities in the case of DCSBM graphs with medium and high modularities. In particular, for each value of graph dimension and of group number, when $Q \geq 0.4$, the clusterings obtained by our method give modularity values that are in a very good agreement with the true modularities as well as with modularities obtained by Louvain, which have the best general behavior. We point out that LeadingEigen in many cases, also for large modularities, obtains clustering with smaller values of modularity with respect to the expected ones and our method outperforms it. This general behaviour is confirmed by the VI curves versus modularity, plotted in Figures 7 to 10, where we compare VI of the clustering obtained by the different methods when the expected (true) blocks are considered as reference partition. Near null values of VI correspond to a good agreement between clustering computed by the considered method and the true network blocks. We can see that, for increasing values of modularity (generally larger than 0.4), we obtain clustering in a good agreement with the true one, and also in terms of VI, our results appear generally better than LeadingEigen. Poor results for small modularities graphs are well known in the context of spectral clustering, due to the behavior of the K-means procedure in these cases, indeed our possible future goal is to substitute K-means with more reliable methods for spatial clustering, such as linear ordering.[4,5,10]

5.2 | Results on real-life networks

Here, we discuss first results obtained by application of our method on the real networks summarized in Section 4.2. Since we do not have information on the number of possible blocks, we applied the K-means algorithm requiring a number of clusters depending on a version of the so-called *eigengap heuristic*. It is a widely used heuristic to set the number of possible clusters in spectral clustering, which finds the rationale in the eigenspace perturbation theory[2] and it seems to work well if the clusters are very well defined. Applying to the singular values of our smooth vectors, the same motivation driving the eigengap heuristic, we set $K = argmax_i |\sigma_{i+1} - \sigma_i|$, with σ_i singular values of smooth vectors, as number of blocks for K-means. In Table 2, we report the number of clusters K and the corresponding modularity Q computed by our method exploiting BootCMatch, as well as the results from LeadingEigen and Louvain. We also report *VI* for the first 2 methods, using Louvain as reference clustering.

We note that, for the graphs ct2010 and caidaRouterLevel, our bootstrap AMG is not able to obtain the desired convergence rate within the fixed number of iterations and we obtain a final composite AMG with convergence rate $\varrho \approx 10^{-3}$ and $\varrho \approx 10^{-6}$, respectively. In the cases of immuno, as22july-06 and email-Enron, we need 33, 26, and 39 AMG components

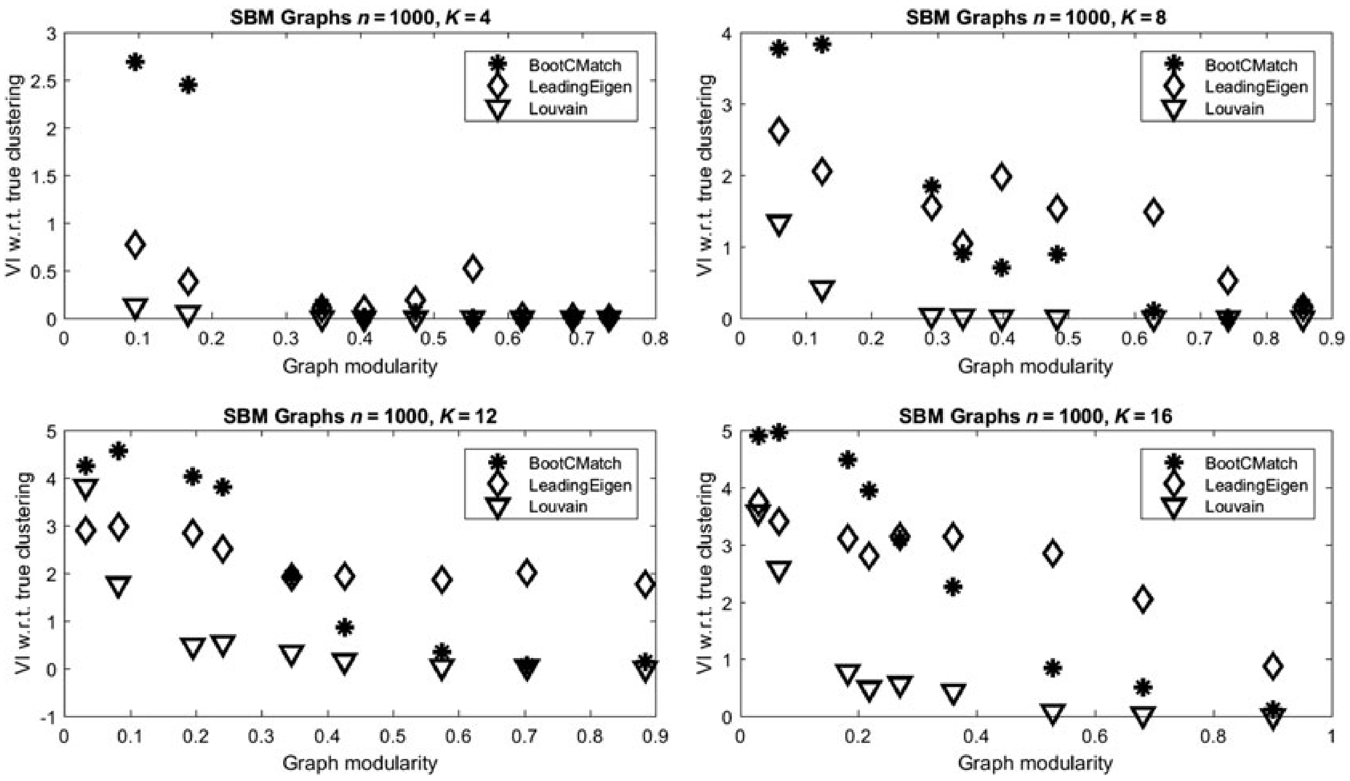

FIGURE 7 Stochastic Block Model graphs ($n = 1000$): Variation of Information (VI) values of different clustering with respect to (w.r.t.) the true clustering

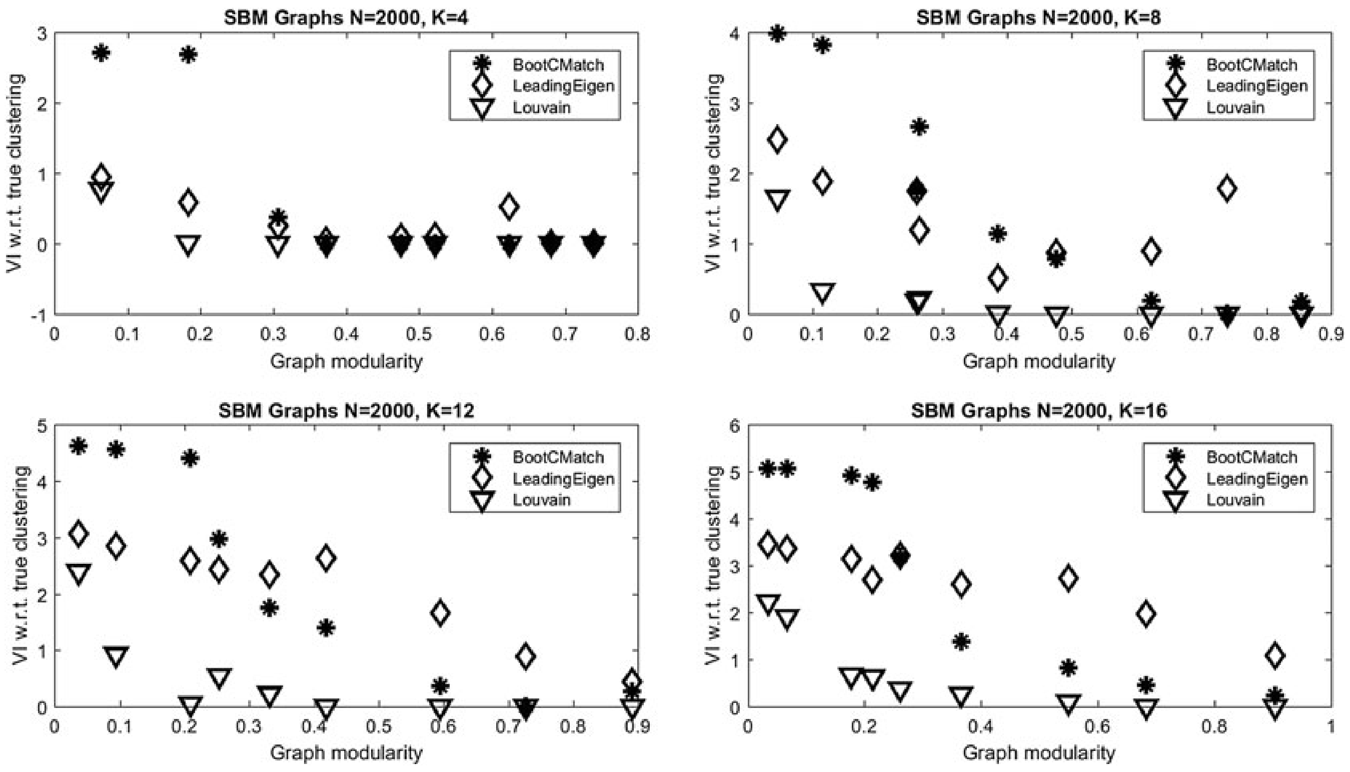

FIGURE 8 Stochastic Block Model (SBM) graphs ($n = 2000$): Variation of Information (VI) values of different clustering with respect to (w.r.t.) the true clustering

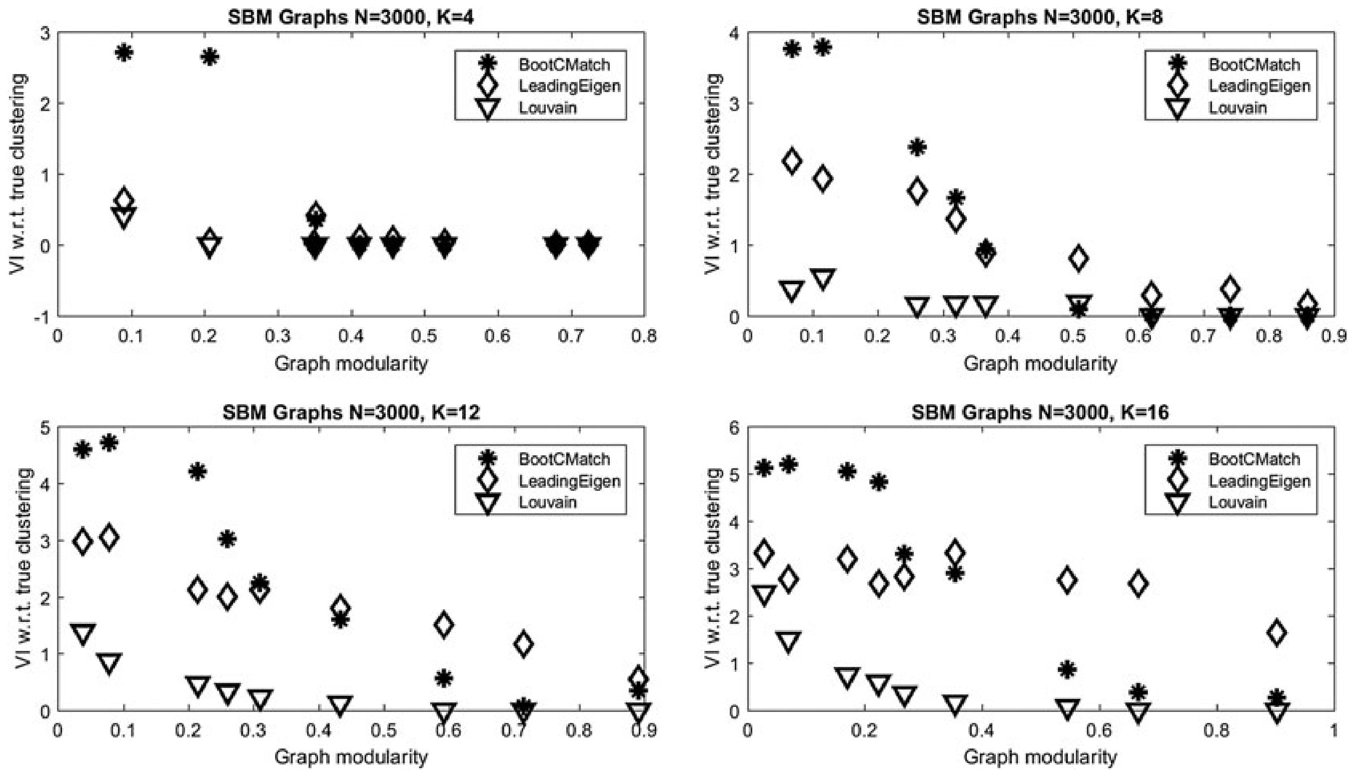

FIGURE 9 Stochastic Block Model (SBM) graphs ($n = 3000$): Variation of Information (VI) values of different clustering with respect to (w.r.t.) the true clustering

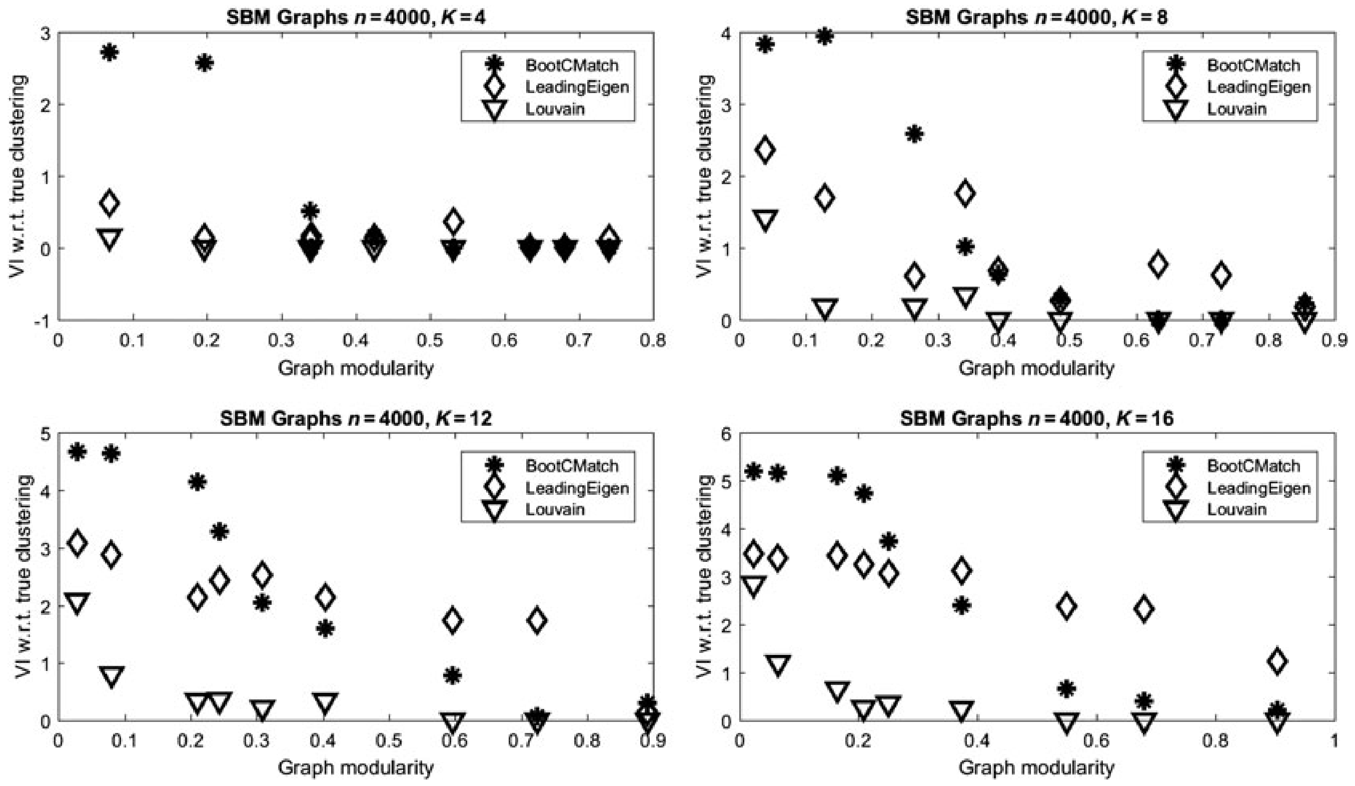

FIGURE 10 Stochastic Block Model (SBM) graphs ($n = 4000$): Variation of Information (VI) values of different clustering with respect to (w.r.t.) the true clustering

TABLE 2 Clustering obtained by different methods: quality measures

	BootCMatch			LeadingEigen			Louvain	
Name	*K*	*Q*	*VI*	*K*	*Q*	*VI*	*K*	*Q*
immuno	21	0.821	1.55	12	0.863	1.03	9	0.826
as22july-06	24	0.431	3.95	4	0.307	3.20	39	0.662
email-Enron	37	0.085	2.67	6	0.394	3.17	291	0.579
ct2010	39	0.954	1.57	20	0.230	4.04	80	0.964
caidaRouterLevel	38	0.315	3.75	2	0.061	4.26	645	0.831

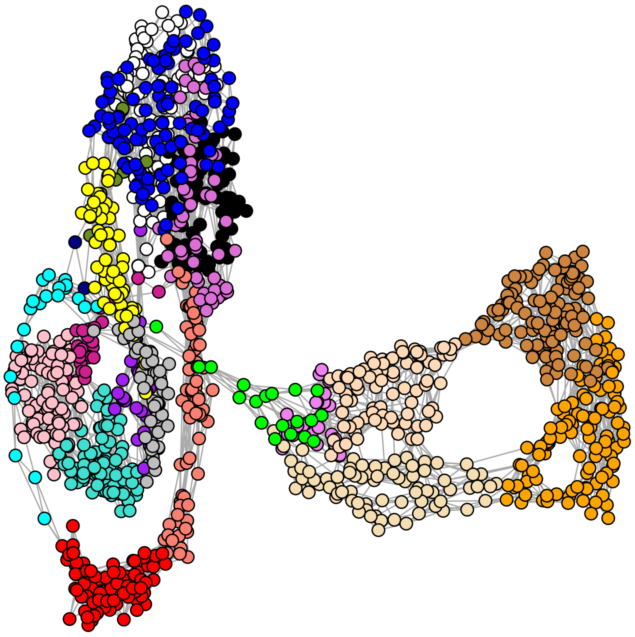
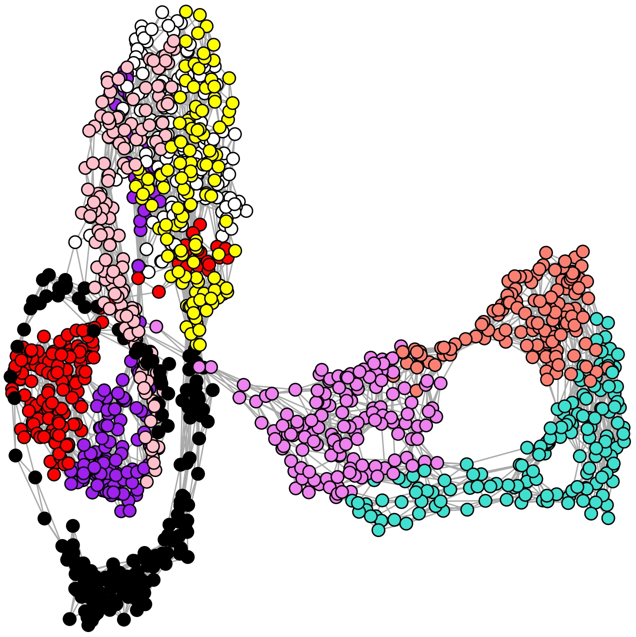

FIGURE 11 Clustering of immuno network. BootCMatch (left) and Louvain (right)

respectively to reach the desired convergence rate. If we compare modularity obtained by our clusterings with the other two methods, we see that, in all the cases but email-Enron, we have a larger value than that obtained by LeadingEigen. Louvain obtains the larger modularities in all the cases and this confirms its good ability to obtain clusterings that maximize modularity; however, we have comparable results in the case of immuno and ct2010 networks, as also confirmed by VI values. In all the cases in which Louvain obtains modularities much larger than our method, we observe a very large number of groups in the Louvain clustering. The analysis of this aspect requires more investigation, since it can be a limitation of the eigengap heuristic used to fix the number of clusters for K-means processing. Furthermore, we observe that the networks on which our method obtains clustering with significantly smaller modularity with respect to the Louvain method have an inhomogeneous distribution of vertex degree. In these cases, the use of normalized Laplacian seems to be more reliable.[2]

The above first results on realistic networks indicate the feasibility of our method as a tool for spectral clustering that overcomes the need to compute Laplacian eigenvectors, giving a linear complexity algorithm to estimate the above eigenvectors with an arbitrary accuracy.

For illustrative purposes, in Figure 11, we visualize the clustering obtained with our method (left) and the clustering obtained by the Louvain method (right) on the immuno network.

6 | CONCLUDING REMARKS

We discussed first results of the application of a bootstrap AMG as a tool for spectral clustering. The results are very promising when dealing with networks with medium and high modularities, ie, when the block structure is well defined. Thus, our method can serve as valid alternative to modularity-based methods, such as the Louvain method, when there is no "ground truth" for better guarantee of results. Our method is also different from more traditional spectral clustering; it exploits approximations of generalized eigenvalue problems (6), where the operator B changes at each bootstrap step; therefore, we have a natural way to decide how many steps to perform, namely, when we reach an operator B that is spectrally equivalent to the original graph Laplacian. For the more standard spectral clustering, it is not as clear how many eigenvectors to choose in advance in order to get satisfactory clustering results. The guarantee that our method provides is a desired property in practice and we view it as an advantage over the more standard spectral clustering approaches. We finally observe that our method has a linear computational complexity (with proportionality constant that grows with the number of vectors used) and can be very competitive for large-scale data sets. Future work includes analysis of the impact of the K-means algorithm on the poor results for small modularities graphs and also of the eigengap heuristic in the case of unknown number of clusters. Furthermore, the exploitation of normalized versions of Laplacian matrix for studying largely irregular networks of increasing size will be considered.

ACKNOWLEDGEMENTS

This work was partially supported by the European Council under the Horizon 2020 Project Energy oriented Centre of Excellence for computing applications - EoCoE, Project ID 676629. The work of the third author was supported in part by the National Science Foundation under grant DMS-1619640. This work was partially performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

ORCID

Pasqua D'Ambra https://orcid.org/0000-0003-2047-4986

REFERENCES

1. Merris R. Laplacian matrices of graphs: a survey. *Linear Algebra Appl*. 1994;197:143-176. https://doi.org/10.1016/0024-3795(94)90486-3
2. von Luxburg U. A tutorial on spectral clustering. *Stat Comput*. 2007;17(4):395-416. https://doi.org/10.1007/s11222-007-9033-z
3. Schaeffer SE. Graph clustering. *Comput Sci Rev*. 2007;1(1):27-64. https://doi.org/10.1016/j.cosrev.2007.05.001
4. Fortunato S. Community detection in graphs. *Physics Reports*. 2010;486(3-5):75-174. https://doi.org/10.1016/j.physrep.2009.11.002
5. Nascimento MCV, de Carvalho ACPLF. Spectral methods for graph clustering - a survey. *Eur J Oper Res*. 2011;211:221-231. https://doi.org/10.1016/j.ejor.2010.08.012
6. Hastie T, Tibshirani R, Friedman J. *The Element of Statistical Learning*. New York, NY: Springer; 2001.
7. R Core Team. An Introduction to R: A Programming Environment for Data Analysis and Graphics. Version 3.5.1. www.r-project.org/about.html. 2018.
8. Meilă M. Comparing clusterings an information based distance. *J Multivar Anal*. 2007;9(5):873-895. https://doi.org/10.1016/j.jmva.2006.11.013
9. Carissimo A, Cutillo L, De Feis I. Validation of community robustness. *Comput Stat Data Anal*. 2018;120:1-24. https://doi.org/10.1016/j.csda.2017.10.006
10. Peng R, Sun H, Zanetti P. Partitioning well-clustered graphs: spectral clustering works. *J Mach Learn Res Workshop Conf Proc*. 2015;40:1-33.
11. Newman MEJ. Community detection in networks: modularity optimization and maximum likelihood are equivalent. *Phys Rev E*. 2016;94(5). Article No. 052315. https://doi.org/10.1103/PhysRevE.94.052315
12. Clauset A, Newman MEJ, Moore C. Finding community structure in very large networks. *Phys Rev E*. 2004;70(6). Article No. 066111. https://doi.org/10.1103/PhysRevE.70.066111
13. Blondel VD, Guillaume JL, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. *J Stat Mech Theory Exp*. 2018;2008(10). Article No. P10008. https://doi.org/10.1088/1742-5468/2008/10/P10008
14. Vassilevski PS. *Multilevel Block Factorization Preconditioners, Matrix-Based Analysis and Algorithms for Solving Finite Element Equations*. New York, NY: Springer; 2008.
15. Ron D, Safro I, Brandt A. Relaxation-based coarsening and multiscale graph organization. *SIAM Multiscale Model Simul*. 2001;9(1):407-423. https://doi.org/10.1137/100791142
16. Chen J, Safro I. Algebraic distance on graphs. *SIAM J Sci Comput*. 2011;33(6):3468-3490. https://doi.org/10.1137/090775087
17. Livne OE, Brandt A. Lean algebraic multigrid (LAMG): fast graph Laplacian linear solver. *SIAM J Sci Comput*. 2012;34(4):B499-B522. https://doi.org/10.1137/110843563
18. Brezina M, Falgout R, MacLachlan S, Manteuffel T, McCormick S, Ruge J. Adaptive smoothed aggregation (αSA). *SIAM J Sci Comput*. 2004;25(6):1896-1920. https://doi.org/10.1137/S1064827502418598
19. Brezina M, Falgout R, MacLachlan S, Manteuffel T, McCormick S, Ruge J. Adaptive algebraic multigrid. *SIAM J Sci Comput*. 2006;27(4):1261-1286. https://doi.org/10.1137/040614402
20. Brandt A, Brannick J, Kahl K, Livshits I. Bootstrap AMG. *SIAM J Sci Comput*. 2011;33(2):612-632. https://doi.org/10.1137/090752973

21. D'Ambra P, Vassilevski PS. Adaptive AMG with coarsening based on compatible weighted matching. *Comput Vis Sci*. 2013;16(2):59-76. https://doi.org/10.1007/s00791-014-0224-9
22. D'Ambra P, Filippone S, Vassilevski PS. BootCMatch: a software package for bootstrap AMG based on graph weighted matching. *ACM Trans Math Softw*. 2018;44-39(25):1-39. https://doi.org/10.1145/3190647
23. Cutillo L, Signorelli M. An inferential procedure for community structure validation in networks. 2017. arXiv:1710.06611.
24. Karrer B, Newman MEJ. Stochastic blockmodels and community structure in networks. *Phys Rev E*. 2011;83. Article No. 016107. https://doi.org/10.1103/PhysRevE.83.016107
25. Newman MEJ. Network Data. http://www-personal.umich.edu/~mejn/netdata/
26. Bader DA, Meyerhenke H, Sanders P. Graph partitioning and graph clustering. In: *Graph Partitioning and Graph Clustering: 10th DIMACS Implementation Challenge Workshop. February 13-14, 2012. Georgia Institute of Technology, Atlanta, GA*. Providence, RI: Center for Discrete Mathematics and Theoretical Computer Science, American Mathematical Society; 2013. *Contemporary Mathematics*; vol. 588.
27. Tarjan R. Depth-first search and linear graph algorithms. *SIAM J Comput*. 1972;1:146-160.
28. Newman MEJ. Finding community structure using the eigenvectors of matrices. *Physical Review*. 2006;974. Article No. 036104. https://doi.org/10.1103/PhysRevE.74.036104

AUTHOR BIOGRAPHIES

Pasqua D'Ambra is a senior scientist at the Institute for Applied Computing of the National Research Council of Italy, Naples branch. She earned her PhD in applied mathematics and computer science from the University of Naples Federico II, Naples, Italy, in 1995. Her research interests are in algorithms and software for parallel numerical linear algebra with applications to fluid dynamics and data analysis, and her activities are carried on in the context of national and international projects, also with PI roles. She is also adjunct professor of numerical computing at the University of Campania, Caserta, Italy, since 2003.

Luisa Cutillo is currently a lecturer in statistics at the University of Leeds, Leeds, UK, and has a very multidisciplinary background. Previously, she was awarded a 2-year Marie S. Curie Individual Fellowship in the research group of Neil Lawrence at the University of Sheffield, Sheffield, UK. Earlier, she was a lecturer at the University of Naples "Parthenope", Naples, Italy, and a member of the Bioinformatics Core of TIGEM in Naples. She received her PhD in applied mathematics from the University of Naples "Federico II", Naples, Italy, and her undergraduate degree in mathematics from the University of Campania "L.Vanvitelli", Caserta, Italy.

Panayot S. Vassilevski is a mathematician in the Center for Applied Scientific Computing at the Lawrence Livermore National Laboratory, Livermore, California, since March 1998. Since 2014, he is also a professor of mathematics at the Portland State University, Portland, Oregon. He earned his PhD from "St. Kliment Ohridski" University of Sofia, Sofia, Bulgaria, in 1984 and worked at the Bulgarian Academy of Sciences, Sofia, Bulgaria, until 1997. His research includes numerical methods for partial differential equations and algorithms for discrete mathematics. He is a Fellow for the Society for Industrial and Applied Mathematics (Class of 2018).

How to cite this article: D'Ambra P, Cutillo L, Vassilevski PS. Bootstrap AMG for spectral clustering. *Comp and Math Methods*. 2019;e1020. https://doi.org/10.1002/cmm4.1020