2013 IEEE 13th International Conference on Data Mining

Spectral Subspace Clustering for Graphs with Feature Vectors

Stephan Günnemann
Carnegie Mellon University, USA
sguennem@cs.cmu.edu

Ines Färber Sebastian Raubach Thomas Seidl
RWTH Aachen University, Germany
{faerber, raubach, seidl}@cs.rwth-aachen.de

***Abstract*—Clustering graphs annotated with feature vectors has recently gained much attention. The goal is to detect groups of vertices that are densely connected in the graph as well as similar with respect to their feature values. While early approaches treated all dimensions of the feature space as equally important, more advanced techniques consider the varying relevance of dimensions for different groups.**

In this work, we propose a novel clustering method for graphs with feature vectors based on the principle of spectral clustering. Following the idea of subspace clustering, our method detects for each cluster an individual set of relevant features. Since spectral clustering is based on the eigendecomposition of the affinity matrix, which strongly depends on the choice of features, our method simultaneously learns the grouping of vertices *and* the affinity matrix. To tackle the fundamental challenge of comparing the clustering structures for different feature subsets, we define an objective function that is unbiased regarding the number of relevant features. We develop the algorithm SSCG and we show its application for multiple real-world datasets.

I. Introduction

Applications producing or handling network data are present in nearly all domains. With the increasing popularity and availability of network data over the last years, its analysis has gained much attention. The task of *graph clustering* is an established mining technique and of interest for the analysis of, e.g., social networks, sensor networks, gene interaction networks, or the web. As for traditional clustering of vector data, the goal of graph clustering is to group similar vertices. Even though there exist multiple definitions of what constitutes a cluster, the common objective is to group the vertices into clusters such that many edges are present *within* each cluster but relatively few edges are existent *between* different clusters.

Besides the mere structural information, in many domains additional information for the objects is available (e.g., feature vectors annotated to the vertices). In a social network, for example, the relationships among people as well as the peoples' individual characteristics such as age or occupation might be given. Analyzing these enriched graphs regarding feature similarity *and* regarding the topological structure is challenging; though, has shown to substantially enhance clustering results [1].

The particular difficulty of this task is that some of the features associated to the vertices might not support or even disagree with the clustering structure. It can be very futile to accommodate, e.g., private music preferences instead of research preferences to the relations shown in a co-authorship network. As known for traditional clustering of vector data, the presence of such noisy or irrelevant features is able to mask the underlying clustering structure. To solve this problem, the paradigm of *subspace clustering* [2] identifies clusters only in the context of their relevant features. Among the graph clustering methods that analyze graphs with feature vectors, only few methods account for irrelevant features [3], [4], [5], [6]. Their experimental evaluation has shown that not necessarily all features exhibit a (strong) correlation with the network and that these non-correlating features can hinder a proper cluster identification.

In this work, we propose a novel clustering method that tackles the problem of irrelevant attributes when clustering graphs with feature vectors by extending the principle of *spectral clustering*. Spectral clustering is an established and widely used clustering paradigm which exploits the ideas of (normalized) graph cuts [7]. It is applicable to graph data as well as vector data and it enjoys great popularity. For spectral clustering the k eigenvectors belonging to the k smallest eigenvalues of, e.g., the graph's normalized Laplacian matrix are used as cluster indicator vectors. In this setting the feature similarity is incorporated into the clustering process only as a weighting for the graphs edges. Bach and Jordan [8] already pointed out that the choice of metric for the similarity structure strongly influences the success of spectral clustering. They furthermore have shown that the presence of irrelevant features has a large impact on the clustering quality and they propose a (semi-)supervised approach to learn a proper affinity matrix.

In this work, we present a fully unsupervised approach for clustering graphs with feature vectors. Our method simultaneously learns the grouping of vertices as well as the affinity matrix used for spectral clustering. We follow the principle of subspace clustering where for each cluster an *individual* set of features might be relevant. Thus, our method excludes locally irrelevant features which hinder the detection of good clustering results. Our contributions are:

- We present a solution for adapting spectral clustering to the problem of subspace clustering for graphs with feature vectors; for each cluster an individual set of relevant features is detected.
- We propose a computation of our objective function that is unbiased w.r.t. the number of relevant features.
- We develop the algorithm SSCG solving our objective.

1550-4786/13 $31.00 © 2013 IEEE
DOI 10.1109/ICDM.2013.110

IEEE computer society

II. Related Work

Our work tackles the problem of clustering graph/network data in the meaning of finding homogeneous sets of vertices in a single graph. This task is also known as community detection or dense subgraph mining [9].

Clustering of graphs with feature vectors. While traditional graph clustering methods concentrate on the mere structural information, there is an increasing interest in also considering complementary information such as vertex features for the clustering process. Methods for this problem setting generally assume that clusters based on the graph's topology and the vertices' features are more meaningful than those based only on one characteristic. In [10] structural and feature information are combined into a single distance function, which can result in clusters with neither a specific graph nor a specific feature pattern. [1] tackles the problem from the features' perspective and extends the k-center problem by an internal connectedness constraint. The authors of [11] use a normalized modularity definition where vertex features are incorporated as edge weights. For minimizing this normalized modularity, a spectral clustering approach is used. [12] introduces a parameter-free approach following the idea of compression. All the above approaches are not able to detect similarities among vertices based on feature subsets. Similar to traditional full-space clustering for vector data, their results will become less meaningful in the presence of irrelevant features. For vector data, the paradigm of subspace clustering [2], [13] reduces the influence of irrelevant features.

Varying relevance of dimensions. Only few approaches were presented so far that attend to the clustering of feature labeled graphs from a subspace clustering perspective. The approach of [3] defines a feature augmented graph, where features are modeled as additional vertices linked to those original vertices showing the specific value for this feature. For the final clusters, objects are only pairwise similar and no particular relevant features can be defined for each cluster. The principle of detecting an individual subset of relevant dimensions for each cluster is so far only fulfilled by three approaches [4], [5], [6]. While [4], [5] exploit the notion of quasi-cliques, which poses strong restrictions regarding the clusters' feature range and their diameter, the method of [6] follows a density based cluster notion for subgraphs as well as for the feature space. The work of [4] generates a huge amount of overlapping clusters leading to high redundancy in the clustering result. To control the level of redundancy, [5], [6] propose models for redundancy handling, introducing additional parameters the user has to specify. Our approach based on spectral clustering determines a partitioning of the vertices and, thus, does not suffer from redundancy.

Spectral clustering. Spectral clustering [7] is suitable for vector data [14], [8] and graph data [11]. Even though the strong influence of the affinity matrix on the clustering result has already been noted, so far no approach exists considering the affinity matrix as part of the unsupervised learning process. [8] confirms the detrimental effect of irrelevant features for spectral clustering. They provide a method for learning the affinity matrix in a (semi-)supervised fashion. Besides assuming a given partition, [8] determines a *global weighting* of the features. As shown for vector data, it might be the case that for different clusters different features prove to be irrelevant. In such cases no feature subset will help to uncover all clusters which makes it desirable to find clusters with *individual sets of relevant features*. Our method simultaneously learns the partition and the underlying affinity matrix, where one important goal is to diminish the influence of irrelevant features for each partition individually.

III. Model

In this section, we present our model for clustering feature labeled graphs in subspace projections. The input for our method is a graph $G{=}(V,E,l)$ with vertices $V{=}\{1,\dots,N\}$, edges $E\subseteq V\times V$ and a labeling function $l:V\to\mathbb{R}^D$ where $Dim{=}\{1,\dots,D\}$ is the set of dimensions. We assume normalized feature vectors in the range $[0,1]$ per dimension.

A. Preliminaries

Among the multitude of different cut definitions, we focus on the frequently used normalized cut because of its strength in avoiding unbalanced cuts [15]. Intuitively, the goal of clustering based on normalized cuts is to find a K-partitioning of the nodes that minimizes the inter-cluster connectivity while at the same time maximizing the intra-cluster connectivity. For this purpose, let $\mathbb{P}^{N,K}$ be the set of all possible (complete and disjoint) K-way partitionings of the set V. It can be represented by the set of binary matrices:

$$\mathbb{P}^{N,K} := \left\{ \mathbf{A}\in\{0,1\}^{N\times K} \mid \sum_{k=1}^{K}\mathbf{a}_k = \mathbf{1}_N \wedge \forall k : \mathbf{a}_k \neq \mathbf{0}_N \right\}$$

where $\mathbf{a}_k$ is the k-th column of the binary matrix $\mathbf{A}$, $\mathbf{1}_N$ is the vector containing only entries equal to 1, and $\mathbf{0}_N$ contains only entries equal to 0. Each $\mathbf{A}\in\mathbb{P}^{N,K}$ represents one possible partitioning and each column vector $\mathbf{a}_k$ stands for one specific group of this partitioning. This definition automatically ensures the orthogonality of the vectors $\mathbf{a}_k$. We denote by $V(\mathbf{a}_k)$ the vertices belonging to group k.

Let $\mathbf{W}=(w_{u,v})_{u,v=1}^{N}$ be the adjacency matrix of a graph G. The cut-value between two groups $V(\mathbf{a}_k)$ and $V(\mathbf{a}_{k'})$ is defined as:

$$cut_{\mathbf{W}}(V(\mathbf{a}_k),V(\mathbf{a}_{k'})) = \sum_{u\in V(\mathbf{a}_k),v\in V(\mathbf{a}_{k'})} w_{u,v} = \mathbf{a}_k^T\cdot\mathbf{W}\cdot\mathbf{a}_{k'}$$

The goal of the normalized cut problem is to find a partitioning $\mathbf{A}\in\mathbb{P}^{N,K}$ that minimizes the following function:

$$nCut_{\mathbf{W}}(\mathbf{A}) = \sum_{k=1}^{K}\frac{\mathbf{a}_k^T\cdot\mathbf{W}\cdot(\mathbf{1}_N-\mathbf{a}_k)}{\mathbf{a}_k^T\cdot\mathbf{W}\cdot\mathbf{1}_N} \qquad (1)$$

The nominator calculates the sum of weights over outgoing edges of cluster k, while the denominator calculates the sum of weights over internal and outgoing edges of cluster k. Thus, the normalized cut trades off low inter-cluster connectivity and high intra-clustering connectivity.

Since optimizing Equation 1 is intractable [15], spectral clustering solves a relaxed problem. By substituting the binary valued cluster indicator vectors $\mathbf{a}_k$ with real-valued vectors, the above problem transforms into a tractable eigenvector problem based on the normalized graph Laplacian $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}$. Here we use $\mathbf{D} = diag(d_1, \ldots, d_k)$ with $d_u = \sum_{v=1}^{N} w_{u,v}$. The K-way partitioning of the nodes can finally be obtained by determining the first K eigenvectors of $\mathbf{L}$, considering each of the N rows as a K-dimensional vector, and by clustering them based on, e.g., k-means. We refer to [7] for more details.

Integrating feature vectors using kernels. Since for the normalized cut and spectral clustering the data is represented by the matrix $\mathbf{W}$, it is easy to incorporate additional aspects into the process of clustering. For example, by applying a kernel transformation $k(x, y)$ on the feature vectors, we can enrich $\mathbf{W}$ by the similarity of these features:

$$w_{u,v} = k(\mathbf{x}, \mathbf{y}) \cdot \mathbb{I}((u, v) \in E) \tag{2}$$

where $\mathbf{x} = l(u)$ and $\mathbf{y} = l(v)$ are the feature vectors of vertices u and v, and $\mathbb{I}$ is the indicator function.

In the following we focus on radial basis function kernels (RBF kernels) where the kernel value $k(\mathbf{x}, \mathbf{y})$ just depends on the norm of $\mathbf{x} - \mathbf{y}$, i.e., $k(\mathbf{x}, \mathbf{y}) = k(\|\mathbf{x} - \mathbf{y}\|)$.[1] Furthermore, in our scenario, it is natural to restrict the consideration to kernels having a non-negative derivative, i.e., $\frac{d}{dx}k(x) \geq 0$ for $x \geq 0$. Thus, increasing the 'dissimilarity' between two feature vectors, decreases the kernel value. These properties hold for a variety of kernels such as, e.g., the Gaussian, Rational quadratic, or Exponential kernel [16].

B. Normalized Cut in Subspace Projections

As mentioned in the introduction, one cannot expect to find clusters in the full dimensional space but in subspace projections of the data. For example, the graph depicted in Figure 1 exhibits no group of vertices being similar w.r.t. all three dimensions. Thus, instead of considering the (unweighted) Euclidean norm between two feature vectors, we use the weighted Euclidean norm

$$\|\mathbf{x}-\mathbf{y}\|_{\mathbf{s}} := \sqrt{(\mathbf{x}-\mathbf{y})^T diag(\mathbf{s})(\mathbf{x}-\mathbf{y})} \;\text{ s.t. } \sum_{i=1}^{D} s_i{=}1, s_i{\geq}0$$

Based on the subspace vector $\mathbf{s}$, we can weight the importance of individual dimensions. Noisy or uninteresting dimensions can be excluded by choosing $s_i \to 0$.

Since we do not know apriori in which subspaces the clusters are located, unlike most approaches, we cannot

[1]To simplify the notations, we will overload the function symbol k. It can either be a binary function or an unary one. The actual use will be clear from the context.

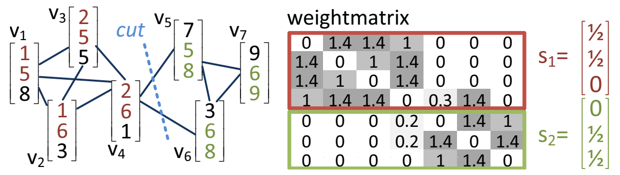

Figure 1. Weight matrix for subspaces s_1 and s_2 belonging to partitioning $\{\{v_1, v_2, v_3, v_4\}, \{v_5, v_6, v_7\}\}$.

assume the matrix $\mathbf{W}$ to be apriori given or static anymore. *In our method, we simultaneously learn the matrix* $\mathbf{W}$ *as well as the object grouping* $\mathbf{A}$.

Let $\mathbf{s}$ be a subspace vector, the matrix $\mathbf{W}_\mathbf{s} = (\hat{w}_{u,v})_{u,v=1}^{N}$ is defined by

$$\hat{w}_{u,v} = k(\|l(u) - l(v)\|_\mathbf{s}) \cdot \mathbb{I}((u, v) \in E) \tag{3}$$

The matrix $\mathbf{W}_\mathbf{s}$ represents the graph when projected to a *single* subspace. Consequently, it corresponds to a global dimensionality reduction. We, however, are interested in finding locally relevant subspaces: each cluster is associated with an *individual* subspace. In our toy example of Figure 1, for the group $\{v_1, v_2, v_3, v_4\}$ features 1 and 2 are relevant, while for group $\{v_5, v_6, v_7\}$ features 2 and 3 are interesting. Thus, instead of considering a single subspace vector $\mathbf{s}$, we are interested in finding a matrix $\mathbf{S} \in [0, 1]^{D \times K}$, where each column represents a (possibly different) subspace vector. Technically, it has to hold $\mathbf{S} \in \mathbb{S}^{D,K}$ where

$$\mathbb{S}^{D,K} := \{\mathbf{S} \in [0, 1]^{D \times K} \mid \forall k : \|\mathbf{s}_k\|_1 = 1\}$$

and $\mathbf{s}_k$ denotes the k-th column of $\mathbf{S}$.

Using individual subspaces introduces a further challenge. The weights between the vertices do not solely depend on $\mathbf{S}$ but they depend on $\mathbf{A}$, too. What is the weight between two vertices u and v? In the case that both nodes belong to the k-th cluster, $u, v \in V(\mathbf{a}_k)$, it is clear that the weight is determined by $(\mathbf{W}_{\mathbf{s}_k})_{u,v}$. Though, how to handle the case where the nodes belong to different clusters? A principled answer to this question can be given by exploiting the relation between random walks and the normalized cut [17].

Considering the graph as a Markov Chain, the problem of minimizing the normalized cut can be interpreted as finding a partitioning such that a random walk (using the stationary distribution of the Markov Chain) stays long within the same cluster and seldom moves between clusters. More technically, in [17] the following is shown: Let $Pr[A \to B|A]$ denote the probability of the random walk to transition from vertex set A to vertex set B in a single step *given that the walk starts in a vertex from* A. Then, the normalized cut is equal to $\sum_{k=1}^{K} Pr[O_k \to (V \backslash O_k)|O_k]$ where O_k denotes the vertices belonging to the k-th cluster.

The *conditional* probability in the equation above provides us with the answer to our original question: Let $u \in V(\mathbf{a}_k)$ and $v \in V(\mathbf{a}_{k'})$, the weight between vertex u and v is $(\mathbf{W}_{\mathbf{s}_k})_{u,v}$, while the weight between v and u is $(\mathbf{W}_{\mathbf{s}_{k'}})_{v,u}$. One has to condition on the subspace $\mathbf{s}_i$ where the random walk starts. The effects can be nicely observed for the weight matrix of Figure 1 (for simplicity, we used $k(x) = 1/x$ in the toy example). Here the i-th row contains the weights based on the subspace of the cluster the vertex v_i belongs to. Thus, in the fourth row, for example, the subspace $\mathbf{s}_1$ is used, while in the fifth row $\mathbf{s}_2$. As a consequence, inter-cluster edges (e.g., (v_4, v_5) and (v_4, v_6)) might have different weights in different rows. Intuitively, when measuring the 'goodness' of cluster k, we project the whole graph to the subspace $\mathbf{s}_k$ and analyze how well cluster k is separated. From the k-th cluster's point of view it does not matter how well it is separated in other subspaces $\mathbf{s}_{k'}$.

Summarizing, given the subspace matrix $\mathbf{S}$ and the object groupings $\mathbf{A}$, the weight matrix is formalized as:

Definition 1: *Subspace Dependent Weight Matrix*
Let $\mathbf{S} \in \mathbb{S}^{D,K}$ be a matrix representing K subspace vectors and $\mathbf{A} \in \mathbb{P}^{N,K}$ a K-way partitioning. The subspace dependent weight matrix is defined as

$$\mathbf{W}_{\mathbf{S},\mathbf{A}} = \sum_{k=1}^{K} \mathbf{W}_{\mathbf{s}_k} \circ \left(\mathbf{a}_k \cdot \mathbf{1}_N^T\right)$$

where $\mathbf{s}_k$ ($\mathbf{a}_k$) is the k-th column of $\mathbf{S}$ ($\mathbf{A}$), $\mathbf{W}_{\mathbf{s}_k}$ as defined in Equation 3, and $\circ$ is the Hadamard product.

By using the term $\mathbf{a}_k \cdot \mathbf{1}_N^T$, we ensure that the whole graph is projected to the subspace $\mathbf{s}_k$ when considering the vertices $V(\mathbf{a}_k)$. Note that in general the weight matrix $\mathbf{W}_{\mathbf{S},\mathbf{A}}$ might not be symmetric even if the underlying graph is undirected (see Figure 1). However, assuming an undirected graph, the matrix shows a certain kind of (a)symmetry: each subblock of $\mathbf{W}_{\mathbf{S},\mathbf{A}}$ induced by a single cluster k, i.e., each block $\mathbf{a}_k \cdot \mathbf{a}_k^T$, is symmetric. This follows since all objects of the same cluster are projected to the same subspace. The asymmetry of $\mathbf{W}_{\mathbf{S},\mathbf{A}}$ is only caused by edges connecting different clusters.

Now we are ready to formalize our overall objective. Our goal is to find a partitioning $\mathbf{A}$ and individual subspaces $\mathbf{S}$ such that the normalized cut based on the matrix $\mathbf{W}_{\mathbf{S},\mathbf{A}}$ is minimized. Thus, we simultaneously optimize multiple objectives: a) we maximize the intra-cluster connectivity and the intra-cluster similarity of the feature vectors w.r.t. the individually selected subspaces b) we minimize the inter-cluster connectivity and inter-cluster similarity of the feature vectors in the corresponding subspaces.

Definition 2: *Minimum Normalized Subspace Cut*
Given the graph $G = (V, E, l)$ and the number of clusters K, the minimum normalized subspace cut (MNSC) is the problem of finding $\mathbf{S}^* \in \mathbb{S}^{D,K}$, $\mathbf{A}^* \in \mathbb{P}^{N,K}$ such that

$$(\mathbf{A}^*, \mathbf{S}^*) = \underset{\mathbf{A}\in\mathbb{P}^{N,K}, \mathbf{S}\in\mathbb{S}^{D,K}}{\arg\min} \{NSCut(\mathbf{A}, \mathbf{S})\}$$

$$\text{where} \quad NSCut(\mathbf{A}, \mathbf{S}) := \sum_{k=1}^{K} \frac{\mathbf{a}_k^T \cdot \mathbf{W}_{\mathbf{S},\mathbf{A}} \cdot (\mathbf{1}_N - \mathbf{a}_k)}{\mathbf{a}_k^T \cdot \mathbf{W}_{\mathbf{S},\mathbf{A}} \cdot \mathbf{1}_N}$$

C. Subspace Unbiased Cut Computation

When considering the *unweighted* Euclidean norm, the distance between objects increases with increasing subspace dimensionality. Thus, comparing the cut-value in, e.g., a 1-dim. subspace with the cut-value in a 5-dim. subspace is not revealing at all. Since our goal, however, is to pick the best subspace among all possible subspaces, we have to realize a fair comparison of the cut values. The cut values should not be biased to specific dimensionalities of the subspaces.

While solutions to this problem have been proposed in the subspace clustering community (e.g., [18]), in our scenario two aspects are worth mentioning: (1) We consider the *normalized* cut. Computing the fraction of cut values might appear to be unbiased; though, it is not. Consider, e.g., the case that the *second* derivate of the kernel function is negative (e.g., the Gaussian and Exponential kernel). In this case, the cut would be biased to low-dimensional clusters. The reason is that the kernel values drop quicker for small norm values than for larger ones. Thus, informally, when adding a further dimension, the edges within a cluster (high feature similarity) loose much of their kernel value while the edges between clusters (dissimilar features) receive almost the same value. Thus, we loose discrimination power between the inter-cluster and intra-cluster edges. (2) We consider the *weighted* Euclidean norm where the weights have to sum up to 1. However, even in this case we observe an increase of the distance values with increasing dimensionality. Thus, overall, also for our objective function we have to realize an unbiased computation.

Our solution. A simple solution to avoid dimensionality bias would be to introduce a regularization parameter that controls the sparsity/density of the vectors $\mathbf{s}_k$. This frequently used principle of regularization, however, is rather ad hoc and introduces additional regularizations parameters which are often hard to set. We do not want to introduce any additional parameters. Instead, we extend the results known in the area of subspace clustering [18].

The basic idea of our principle is to adapt the computation of the norm such that we obtain an unbiased estimation, i.e., the expected distance (and its variance) between the feature vectors should be constant and, thus, independent of the selected subspace. For this purpose, we first formalize:

Definition 3: *Unbiased parametric family*
Given a parametric family $\mathcal{F} = \{f_\mathbf{s} \mid \mathbf{s} \in \Theta\}$ of functions $f_\mathbf{s} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ and a (multidimensional) probability density function τ over $\mathbb{R}^d$. $\mathcal{F}$ is called unbiased if

$$\forall \mathbf{s}, \mathbf{s}' \in \Theta : \mathrm{E}\left[f_\mathbf{s}(X, Y)\right] = \mathrm{E}\left[f_{\mathbf{s}'}(X, Y)\right] < \infty$$
$$\forall \mathbf{s}, \mathbf{s}' \in \Theta : \mathrm{Var}\left[f_\mathbf{s}(X, Y)\right] = \mathrm{Var}\left[f_{\mathbf{s}'}(X, Y)\right] < \infty$$

where X and Y are i.i.d. with $X \sim \tau, Y \sim \tau$

The family $\mathcal{F}$ might, for example, be the set of functions computing the weighted Euclidean norm, i.e., we would have $f_\mathbf{s}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_\mathbf{s}$ where Θ consists of all valid subspace vectors. The probability density function τ corresponds to the null model the feature vectors are generated from. Intuitively, it is the distribution when expecting no clusters in the data. Based on our setting, it corresponds to the uniform distribution over the hypercube $[0,1]^D$, i.e. $\tau(\mathbf{x}) = 1$ if $\mathbf{x} \in [0,1]^D$, 0 else.

If a family $\mathcal{F}$ of functions is unbiased, we can do a fair comparison between the function values of $f_{\mathbf{s}}(\mathbf{x},\mathbf{y})$ and $f_{\mathbf{s}'}(\mathbf{x},\mathbf{y})$. As mentioned above, the weighted Euclidean norm is not unbiased. Though, we can show the following:

Theorem 1*: Let Θ_D be the set of all possible D-dimensional subspace vectors and*

$$\rfloor\!\lceil \mathbf{x}-\mathbf{y} \rceil\!\lfloor_{\mathbf{s}} := \frac{\|\mathbf{x}-\mathbf{y}\|_{\mathbf{s}} - \mathrm{E}\left[\|\mathbf{x}-\mathbf{y}\|_{\mathbf{s}}\right]}{\sqrt{\mathrm{Var}\left[\|\mathbf{x}-\mathbf{y}\|_{\mathbf{s}}\right]}} + \min_{\mathbf{s}' \in \Theta_D} \frac{\mathrm{E}\left[\|\mathbf{x}-\mathbf{y}\|_{\mathbf{s}'}\right]}{\sqrt{\mathrm{Var}\left[\|\mathbf{x}-\mathbf{y}\|_{\mathbf{s}'}\right]}}$$

The parametric family $\mathcal{F}=\{\rfloor\!\lceil \mathbf{x}-\mathbf{y} \rceil\!\lfloor_{\mathbf{s}} \mid \mathbf{s} \in \Theta_D\}$ is unbiased.

Proof: Using the abbreviations $\mu_{\mathbf{s}}:=\mathrm{E}\left[\|\mathbf{x}-\mathbf{y}\|_{\mathbf{s}}\right]$ and $\sigma_{\mathbf{s}}:=\sqrt{\mathrm{Var}\left[\|\mathbf{x}-\mathbf{y}\|_{\mathbf{s}}\right]}$, then for all $\mathbf{s} \in \Theta_D$ it holds

• $\mathrm{E}\left[\rfloor\!\lceil \mathbf{x}-\mathbf{y} \rceil\!\lfloor_{\mathbf{s}}\right] = \frac{1}{\sigma_{\mathbf{s}}}(\mathrm{E}\left[\|\mathbf{x}-\mathbf{y}\|_{\mathbf{s}}\right] - \mu_{\mathbf{s}}) + \min_{\mathbf{s}' \in \Theta_D} \frac{\mu_{\mathbf{s}'}}{\sigma_{\mathbf{s}'}} = \min_{\mathbf{s}' \in \Theta_D} \frac{\mu_{\mathbf{s}'}}{\sigma_{\mathbf{s}'}} = c_1$

• $\mathrm{Var}\left[\rfloor\!\lceil \mathbf{x}-\mathbf{y} \rceil\!\lfloor_{\mathbf{s}}\right] = \mathrm{E}\left[(\rfloor\!\lceil \mathbf{x}-\mathbf{y} \rceil\!\lfloor_{\mathbf{s}} - c_1)^2\right] = \mathrm{E}\left[\left(\frac{\|\mathbf{x}-\mathbf{y}\|_{\mathbf{s}} - \mu_{\mathbf{s}}}{\sigma_{\mathbf{s}}}\right)^2\right] = \frac{1}{\sigma_{\mathbf{s}}^2} \cdot \mathrm{E}\left[(\|\mathbf{x}-\mathbf{y}\|_{\mathbf{s}} - \mu_{\mathbf{s}})^2\right] = \frac{1}{\sigma_{\mathbf{s}}^2} \cdot \mathrm{Var}\left[\|\mathbf{x}-\mathbf{y}\|_{\mathbf{s}}\right] = \frac{1}{\sigma_{\mathbf{s}}^2} \cdot \sigma_{\mathbf{s}}^2 = 1 = c_2$

Since c_1 and c_2 are independent from $\mathbf{s} \in \Theta_D$, the parametric family $\mathcal{F}$ is unbiased. ■

Intuitively, $\rfloor\!\lceil . \rceil\!\lfloor_{\mathbf{s}}$ is the z-score normalized version of $\|.\|_{\mathbf{s}}$. Thus, instead of measuring the absolute norm between two features, we measure the deviation to the expected value. Since $\rfloor\!\lceil . \rceil\!\lfloor_{\mathbf{s}}$ is guaranteed to be non-negative, it is possible to replace the value of $\|.\|_{\mathbf{s}}$ in Equation 3 by the unbiased measure $\rfloor\!\lceil . \rceil\!\lfloor_{\mathbf{s}}$. The question remains whether $k(\rfloor\!\lceil \mathbf{x}-\mathbf{y} \rceil\!\lfloor_{\mathbf{s}})$ still corresponds to a valid kernel transformation [16]. While in general the use of $\rfloor\!\lceil . \rceil\!\lfloor_{\mathbf{s}}$ does not lead to a valid Mercer kernel, we can show:

Theorem 2*: Given the exponential kernel $k_\theta(t) = e^{-\frac{t}{\theta}}$ with scaling parameter θ. When solving the MNSC problem, replacing $\|.\|_{\mathbf{s}}$ in Equation 3 by $\rfloor\!\lceil . \rceil\!\lfloor_{\mathbf{s}}$ is equivalent to using the (original!) norm $\|.\|_{\mathbf{s}}$ in combination with the exponential kernel based on the scaling parameter $\theta_{\mathbf{s}} := \theta \cdot \sqrt{\mathrm{Var}\left[\|\mathbf{x}-\mathbf{y}\|_{\mathbf{s}}\right]}$.*

Proof: Let $\widetilde{\mathbf{W}}_{\mathbf{s}}$ be the weight matrix according to Eq. 3 using the (unbiased but potentially invalid) kernel values $k_\theta(\rfloor\!\lceil . \rceil\!\lfloor_{\mathbf{s}})$, and let $\mathring{\mathbf{W}}_{\mathbf{s}}$ be the weight matrix using the (valid) kernel values $k_{\theta_{\mathbf{s}}}(\|.\|_{\mathbf{s}})$. We show that using $\widetilde{\mathbf{W}}_{\mathbf{s}}$ is equivalent to using $\mathring{\mathbf{W}}_{\mathbf{s}}$ when solving the MNSC problem.

• We first reformulate the objective function: Since $\mathbf{A}$ is a partitioning, for all k, k' with $k \neq k'$ it holds $\mathbf{a}_k^T \cdot \left(\mathbf{a}_{k'} \cdot \mathbf{1}_N^T\right) = \mathbf{0}_N^T$. Thus, the objective function in Def. 2 can be written as

$$NSCut(\mathbf{A},\mathbf{S}) = \sum_{k=1}^{K} \frac{\mathbf{a}_k^T \cdot \left(\sum_{k'=1}^{K} \mathbf{W}_{\mathbf{s}_{k'}} \circ \left(\mathbf{a}_{k'} \cdot \mathbf{1}_N^T\right)\right) \cdot (\mathbf{1}_N - \mathbf{a}_k)}{\mathbf{a}_k^T \cdot \left(\sum_{k'=1}^{K} \mathbf{W}_{\mathbf{s}_{k'}} \circ \left(\mathbf{a}_{k'} \cdot \mathbf{1}_N^T\right)\right) \cdot \mathbf{1}_N}$$

$$= \sum_{k=1}^{K} \frac{\mathbf{a}_k^T \cdot (\mathbf{W}_{\mathbf{s}_k} \circ (\mathbf{a}_k \cdot \mathbf{1}_N^T)) \cdot (\mathbf{1}_N - \mathbf{a}_k)}{\mathbf{a}_k^T \cdot (\mathbf{W}_{\mathbf{s}_k} \circ (\mathbf{a}_k \cdot \mathbf{1}_N^T)) \cdot \mathbf{1}_N} = \sum_{k=1}^{K} \frac{\mathbf{a}_k^T \cdot \mathbf{W}_{\mathbf{s}_k} \cdot (\mathbf{1}_N - \mathbf{a}_k)}{\mathbf{a}_k^T \cdot \mathbf{W}_{\mathbf{s}_k} \cdot \mathbf{1}_N}$$

Here, $\mathbf{W}_{\mathbf{s}_k}$ is used as a placeholder for either $\widetilde{\mathbf{W}}_{\mathbf{s}_k}$ or $\mathring{\mathbf{W}}_{\mathbf{s}_k}$.

• Using the abbreviations from the proof of Theorem 1, the kernel functions can be reformulated as

$$k_\theta(\rfloor\!\lceil . \rceil\!\lfloor_{\mathbf{s}}) = e^{-\frac{\frac{\|.\|_{\mathbf{s}} - \mu_{\mathbf{s}}}{\sigma_{\mathbf{s}}} + c_1}{\theta}} = e^{\frac{\mu_{\mathbf{s}}}{\sigma_{\mathbf{s}} \cdot \theta} - \frac{c_1}{\theta}} \cdot e^{-\frac{\|.\|_{\mathbf{s}}}{\sigma_{\mathbf{s}} \cdot \theta}} = c_{\mathbf{s}} \cdot k_{\theta_{\mathbf{s}}}(\|.\|_{\mathbf{s}})$$

where $c_{\mathbf{s}} := e^{\frac{\mu_{\mathbf{s}}}{\sigma_{\mathbf{s}} \cdot \theta} - \frac{c_1}{\theta}}$ is a constant depending on the subspace s. It follows: $\widetilde{\mathbf{W}}_{\mathbf{s}} = c_{\mathbf{s}} \cdot \mathring{\mathbf{W}}_{\mathbf{s}}$

• Plugging this result into the reformulated objective function, we get: $\sum_k \frac{\mathbf{a}_k^T \cdot \widetilde{\mathbf{W}}_{\mathbf{s}_k} \cdot (\mathbf{1}_N - \mathbf{a}_k)}{\mathbf{a}_k^T \cdot \widetilde{\mathbf{W}}_{\mathbf{s}_k} \cdot \mathbf{1}_N} = \sum_k \frac{\mathbf{a}_k^T \cdot c_{\mathbf{s}_k} \cdot \mathring{\mathbf{W}}_{\mathbf{s}_k} \cdot (\mathbf{1}_N - \mathbf{a}_k)}{\mathbf{a}_k^T \cdot c_{\mathbf{s}_k} \cdot \mathring{\mathbf{W}}_{\mathbf{s}_k} \cdot \mathbf{1}_N} = \sum_k \frac{\mathbf{a}_k^T \cdot \mathring{\mathbf{W}}_{\mathbf{s}_k} \cdot (\mathbf{1}_N - \mathbf{a}_k)}{\mathbf{a}_k^T \cdot \mathring{\mathbf{W}}_{\mathbf{s}_k} \cdot \mathbf{1}_N}$ $\Rightarrow$ using $\widetilde{\mathbf{W}}_{\mathbf{s}}$ or $\mathring{\mathbf{W}}_{\mathbf{s}}$ is equivalent. ■

Thus, using $\rfloor\!\lceil . \rceil\!\lfloor_{\mathbf{s}}$ will still lead to a valid Mercer kernel. Furthermore, to realize an unbiased computation of the cut, we can simply adapt the scaling parameter of the exponential kernel. We actually do not have to compute $\rfloor\!\lceil . \rceil\!\lfloor_{\mathbf{s}}$; particularly, the term $\mathrm{E}\left[\|\mathbf{x}-\mathbf{y}\|_{\mathbf{s}}\right]$ vanishes completely.

Overall, when using the exponential kernel, we can obtain a subspace unbiased cut computation while still preserving the properties of a valid Mercer kernel. Interestingly, the exponential kernel does not only show these nice theoretical properties, it also has outperformed all other kernels in our empirical studies (Figure 2). Unlike many methods with artificial regularization parameters, our subspace unbiased cut computation does not introduce any additional parameters.

D. Complexity Analysis

Statements like "the normalized cut is NP-complete" have to be regarded carefully. Since usually these problems are formulated as optimization problems, the classical definitions of complexity classes – which were designed for decision problems – cannot be applied straightforward [19]. Thus, we consider the decision problem version of MNSC.

Definition 4*: Decision problem version of MNSC The decision problem version to the MNSC optimization problem is: Given a graph $G = (V,E,l)$, the number of groups K, and a constant C. Is there a normalized subspace cut with value $\leq C$?*

Theorem 3*: The decision problem version of MNSC is* NP-complete *if the kernel function can be evaluated in polynomial time w.r.t. G, K, and C.*

Proof: a) MNSC is NP-hard: We provide a polynomial reduction of the 'usual' normalized cut problem (which does not handle feature vectors) to MNSC, i.e. NCUT $\leq_P$ MNSC.

• Input mapping: The input $G{=}(V,E)$ of NCUT is mapped to an input $G'{=}(V',E',l')$ of MNSC with $V'{=}V$, $E'{=}E$ and $l'(v){=}0$ for all $v \in V$ (i.e. each node has the same feature vector). This transformation can be done in polynomial time.

• MNSC generates a valid NCUT solution: Let $\mathbf{X}$ denote the adjacency matrix used for NCUT. Since all feature vectors in G' are identical, it holds $\mathbf{W}_{\mathbf{S},\mathbf{A}} = k(0) \cdot \mathbf{X}$. The constant value $k(0)$ does not affect the resulting optimal cut. Thus, the solution of MNSC corresponds to the solution of NCUT. Since NCUT is NP-complete [15], MNSC is NP-hard.

b) MNSC is in NP: We use the verifier-based definition of NP. Given a certificate ($\mathbf{A}$,$\mathbf{S}$), we prove that its correctness,

i.e., the equation $NSCut(\mathbf{A}, \mathbf{S}) \leq C$, can be verified in *polynomial time*. The following complexities hold:
• computing the norm: $T_n := \mathcal{O}(d)$
• computing the kernel: $T_k := \mathcal{O}(p)$, with p is a polynomial
• computing $\mathbf{W}_{\mathbf{S},\mathbf{A}}$ (Def. 1): $T_w := \mathcal{O}(|E| \cdot (T_n + T_k))$
• computing $NSCut$ (Def. 2): $T_c := \mathcal{O}(k \cdot |E|)$
For T_w and T_c we exploited the sparsity of the weight matrix and the fact that each vertex belongs to a single cluster. Overall, computing $NSCut(\mathbf{A}, \mathbf{S})$ is in class $\mathcal{O}(|E| \cdot (d + p + k)) \in$ P. Since the verification is in P, MNSC is in NP.

c) combining a) and b) $\Rightarrow$ MNSC is NP-complete. ■

Thus, even though our model uses an adaptive $\mathbf{W}$ depending on $\mathbf{S}$ and $\mathbf{A}$, its complexity class is identical to the one of the normalized cut where $\mathbf{W}$ is assumed to be static. However, two aspects have to be noted: First, the input for our model is more complex since we consider feature-labeled graphs. Second, these results apply for the decision problem version. It does not necessarily follow that the optimization problems are equally complex, i.e., the degrees of approximability might be different [19]. This study is left for future work.

IV. Algorithm

As shown by Theorem 3, we cannot expect to find an efficient algorithm computing an exact solution to the MNSC problem. Alternatively, we design an algorithm computing an approximate solution based on the following observations: (1) Keeping the matrix $\mathbf{W}$ fix, determining the optimal partitioning $\mathbf{A}$ is independent of $\mathbf{S}$ and reduces to the traditional normalized cut problem. (2) As shown in the proof of Theorem 2, the objective function can be written as

$$\sum_{k=1}^{K} g_{\mathbf{a}_k}(\mathbf{s}_k) \quad \text{with} \quad g_{\mathbf{a}}(\mathbf{s}) := \frac{\mathbf{a}^T \cdot \mathbf{W}_{\mathbf{s}} \cdot (\mathbf{1}_N - \mathbf{a})}{\mathbf{a}^T \cdot \mathbf{W}_{\mathbf{s}} \cdot \mathbf{1}_N}$$

Thus, if the matrix $\mathbf{A}$ is given, the subspaces $\mathbf{s}_k$ can be optimized for each cluster *independently*. This independence drastically reduces the hardness. Since for given $\mathbf{A}$ and $\mathbf{S}$, the matrix $\mathbf{W}$ is completely determined, these observations naturally lead to an iterative algorithm where we optimize one variable while keeping the others fix. Such a procedure of alternating optimization is well established for many tasks. Our method works as follows:

```
1: initialize $\mathbf{W}^{(0)}$
2: for($t = 1, \ldots$)
3:     compute normalized Laplacian $\mathbf{L}^{(t)} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}^{(t-1)}$
4:     compute first $k$ eigenvectors $u_1, \ldots, u_k$ of $\mathbf{L}^{(t)}$
5:     determine $\mathbf{A}^{(t)}$ by performing k-means clustering
       on $U = [u_1, \ldots, u_k] \in \mathbb{R}^{N \times k}$
6:     determine $\mathbf{S}^{(t)} = [\mathbf{s}_1, \ldots, \mathbf{s}_k]$ by minimizing $g_{\mathbf{a}_i}(\mathbf{s}_i)$
       (can be done for each cluster $i$ independently)
7:     compute $\mathbf{W}^{(t)}$ (cf. Def. 1) based on $\mathbf{A}^{(t)}$ and $\mathbf{S}^{(t)}$
8:     stop if cut-value has converged
```

We continue by briefly discussing each step of the method.

Initialization & update of W. Since a priori no information about the relevance of dimensions is given, we initialize $\mathbf{W}$ (line 1) using non-informative weights, i.e., implicitly each entry of $\mathbf{S}$ is assumed to be $1/D$. Since in this case the subspaces for all clusters are identical, the dependency of $\mathbf{W}$ on $\mathbf{A}$ vanishes. Consequently, we can apply Def. 1 even without the knowledge of an initial partitioning $\mathbf{A}$. For the recomputation of the weight matrix (line 7), the subspaces per cluster might differ and we have to incorporate the actual partitioning $\mathbf{A}$. The complexity of this step is $\mathcal{O}(|E| \cdot D)$ since we have to recompute the weight for each edge, which is dominated by the computation of the norm ($\mathcal{O}(D)$).

Update of A. As mentioned, the update of $\mathbf{A}$ reduces to a traditional normalized cut problem. Note that the matrix $\mathbf{W}$ and, thus, the Laplacian $\mathbf{L}$ are sparse when the underlying graph is sparse (which holds for most real graphs). Thus, we can use efficient sparse eigenvalue solvers. Additionally, multiple techniques to speed up the computation of spectral clustering for large matrices have been proposed (e.g. [14]). All these techniques can be combined with our method.

While spectral clustering has been successfully applied in many applications, it is fair to mention that this relaxation does not provide any theoretical bound on the error of the cut value [7]. That is, in theory, there is no guarantee that the determined cut value is close to the optimal solution. Consequently, the new partitioning might not lower the previously obtained cut value. Thus, it might be useful to add an additional termination criterion to the above algorithm as, e.g., checking whether the best cut value obtained so far has been decreased during the last m iterations or using an optimization scheme based on simulated annealing.

Update of S. Even though updating $\mathbf{s}_k$ can be done for each cluster individually, the function $g_{\mathbf{a}}(\mathbf{s})$ is still hard to minimize since it is neither convex nor concave in $\mathbf{s}$. We analyzed multiple different strategies for finding local minima of this function, such as gradient descent and different greedy approaches. Here, we present only the final solution used for our approach. We selected this principle based on the following observations: a) it is very efficient to compute, b) it has obtained good normalized subspace cut values in a variety of experiments, c) the results allow an intuitive interpretation as required in many application domains. The last aspect is realized as follows: While our general model allows to use arbitrary subspace vectors, for easy interpretation we follow the principle of most traditional subspace clustering approaches [2], [13], where the relevant dimensions show uniform importance (i.e., we consider vectors as $(1/3, 0, 0, 1/3, 1/3)$). Since each dimension is either non-relevant or equally important for a cluster, an easy interpretation is possible. Formally, we consider the set

$$\mathbb{L} = \{\mathbf{s} \in [0,1]^D \mid \left(s_d > 0 \Leftrightarrow s_d = |\{i \in Dim | s_i > 0\}|^{-1}\right)\}$$

Even though the set $\mathbb{L}$ is finite, its size grows exponentially in D. Since it is intractable to enumerate and evaluate all of its members, we restrict to a meaningful subset. To ensure the selection of the most expressive dimensions, we traverse $\mathbb{L}$ starting with low-dimensional subspaces and successively expanding the best subspace with further dimensions:

```
1: $\mathbf{s}_{old} = 0^D$, $d = 0$ // current subspace and dimensionality
2: $\hat{\mathbb{L}} = \{\mathbf{s} \in \mathbb{L} \mid \exists_{=1} x \in Dim : s_x > 0\}$ // 1-d subspaces
3: select subspace $\mathbf{s}^* = \arg\min_{\mathbf{s}\in\hat{\mathbb{L}}}\{g_{\mathbf{a}}(\mathbf{s})\}$
4: $\mathbf{s}_{new} = (\mathbf{s}_{old} \cdot d + \mathbf{s}^*)/(d+1)$
5: if $g_{\mathbf{a}}(\mathbf{s}_{new}) < g_{\mathbf{a}}(\mathbf{s}_{best})$ then $\mathbf{s}_{best} = \mathbf{s}_{new}$
6: set $d = d + 1$ and $\hat{\mathbb{L}} = \hat{\mathbb{L}} \backslash \{\mathbf{s}^*\}$
7: goto 3 until $\hat{\mathbb{L}} = \emptyset$
```

This method ranks based on the set $\hat{\mathbb{L}}$, which represents the 1-dimensional subspaces. In line 4, we increase the dimensionality of the subspace vector $\mathbf{s}_{new}$ by 'adding' the next best 1-dimensional subspace s^*. The dimensionality of $\mathbf{s}_{new}$ increases in each iteration; thus, guaranteeing termination. Since the ranking in line 3 needs to be done only once, the function $g_{\mathbf{a}}$ needs to be evaluated only D times. Thus, the overall complexity for updating the subspace of a cluster is $\mathcal{O}(D \cdot t_g) = \mathcal{O}(D^2 \cdot |E|)$, where t_g is the complexity to evaluate the function $g_{\mathbf{a}}(.)$.

V. Experimental Analysis

Setup. We compare SSCG with a variety of other partitioning clustering techniques: OptiComb [11], CoClus [10], SA-Clustering [3], and PICS [12] are methods using graph and feature information. We denote with SpectGraph, traditional spectral clustering using only the structural information. SpectVec1&2 is spectral clustering using only feature information. The first one uses the complete similarity graph, the second one uses the kNN similarity graph (cf. [7]). Proclus [13] is a subspace clustering technique for vector data. The number of clusters K and the scaling parameter θ are chosen identical for each method. Since SA-Clustering can only handle categorical data, for this method we discretized each numerical dimension into 10 bins. Accordingly, for PICS, which handles only binary data, numerical data is discretized into two bins. To ensure a fair evaluation, we only consider partitioning clustering approaches in this work. Since overlapping clustering approaches as, e.g., [4], [5] follow a completely different objective, a comparison with the methods mentioned above would always be biased to one of the paradigms. All experiments were conducted on 2.3 GHz Opteron CPUs with Java6 64bit.

For case studies on real world data, we use graphs extracted from the DBLP database, the arXiv database, the Internet Movie Database, the German soccer premier league, patent data, and gene interaction networks. We provide all datasets and their descriptions on our website.[2] Furthermore, we generate synthetic data based on the planted partitions model [20]. Intuitively, given the desired number of clusters and the vertices belonging to each cluster, we randomly add edges between and within clusters according to a specified density. To generate the feature vectors, given the overall dimensionality, we randomly select a given number of relevant dimensions for each cluster. For each cluster an individual set of dimensions is used. By default we generate 20 dimensional data with 10 clusters, each with 100 vertices and 10 relevant features. The average density is 0.4.

[2]http://dme.rwth-aachen.de/SSCG

For synthetic data, clustering quality is measured via the F1 value [21]. For real world data, where no ground truth is given, we use internal evaluation metrics: the normalized subspace cut (NSCut), the usual NCut considering only graph information, and the within cluster sum of squares (total distance, TD) considering only the feature information. To ensure comparability, the internal measures are always computed w.r.t. the input graph (since some approaches perform graph transformations).

Comparison on Synthetic Data. We start by analyzing the effect of different kernels (Fig. 2). Since different kernels lead to different cut values, comparing the obtained cut is unfair. Instead, we compare the clustering quality. As shown, the exponential kernel (leftmost bars) leads to the highest quality and simultaneously obtains the lowest runtime. Thus, besides being theoretically sound (Sec. III-C), the exponential kernel also empirically performs best. It is used for all further experiments.

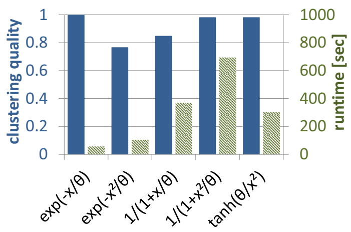

Figure 2. Effect of kernels

Even though our focus is on evaluating the clustering quality of SSCG, we briefly analyze the methods' efficiency. In Fig. 3 (left) we increase the number of vertices in the graph. Since most methods use eigenvalue-decomposition, the slopes of their curves are in a similar range. Here, we determined the eigenvectors using QR-decomposition; as mentioned in Sec. IV, more efficient methods can be used instead. Two algorithms differ from the common shape. PICS scales slightly better than the other methods since it does not use eigenvalue decomposition. It is, however, restricted to binary data. Proclus is very efficient but as we will see in Fig. 4 its clustering quality is very low.

In Fig. 3 (right) we increase the dimensionality of the data. The only slight increase of the methods' runtimes indicates

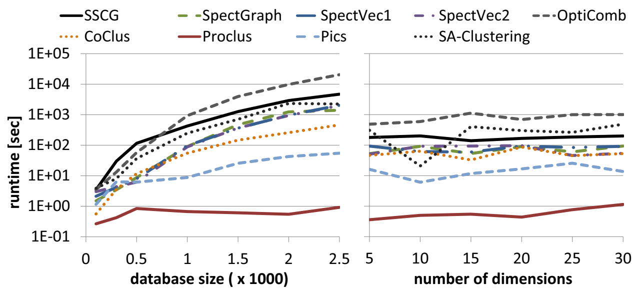

Figure 3. Runtime scalability

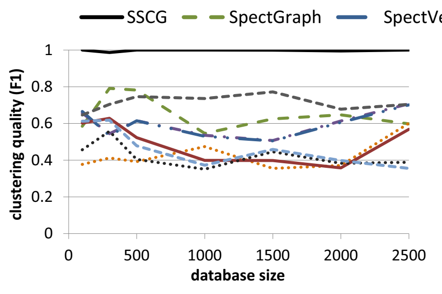

Figure 4. Quality vs. database size

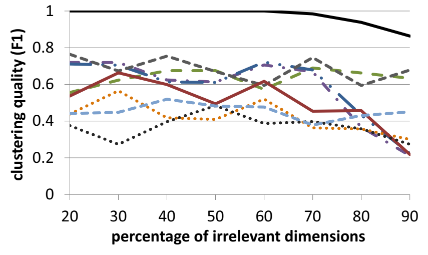

Figure 5. Quality vs. feature noise

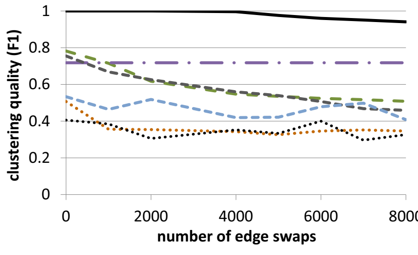

Figure 6. Quality vs. edge noise

that the eigenvalue-decomposition (which is independent of the data's dimensionality) dominates the overall runtime. Our optimization of the subspace is very efficient.

In Fig. 4 we show the methods' clustering quality for an increasing database size. Only SSCG is able to detect the clustering structure. The competing approaches cannot handle data where some features are irrelevant since these features obfuscate the clustering structure in the full-space.

In Fig. 5 we increase the number of irrelevant dimensions per cluster. Starting with almost full-space clusters (16d), we successively lower the clusters' dimensionality (down to 2d clusters). While SSCG shows almost perfect quality, most of the other approaches decrease in their quality; the more irrelevant features the harder to detect the clusters. SpectGraph is not affected by irrelevant features since it only uses the graph structure; though, the absolute quality is low. Interestingly, although involving feature information, OptiComb obtains almost the results of SpectGraph.

While the previous experiment has shown the effects when varying the 'quality' of the feature vectors (i.e., increasing the fraction of irrelevant features), we now analyze the methods' behavior when the structural information is distorted. In Fig. 6 we randomly relocate a certain number of edges. SSCG is only slightly affected. By additionally exploiting the features, the results are almost stable. In contrast, the other graph based methods show a strong decrease. Obviously, the methods using only feature information are not affected (for illustration, only SpectVec2 is plotted).

Overall, SSCG's runtime is comparable to all other methods using spectral clustering and it is the only method simultaneously achieving high clustering qualities even in the presence of many irrelevant features.

Evaluation on Real World Data. Since for real world data no ground truth is given, our following case studies should show two aspects: 1) The clusters found by SSCG are meaningful. We solve this issue by presenting interesting results detected by our method and by analyzing internal characteristics of the clustering result (e.g., the cut values). 2) While not being able to discuss the results of all competitors, we can examine whether a similar result than that of SSCG can already be determined by competing methods. This issue is solved by computing the normalized mutual information ([22], NMI$_{joint}$) between the competitors' results to the one of SSCG. A low NMI value indicates, that SSCG is able to produce novel cluster insights, while not implying that the result of the particular competitor is bad or meaningless. An extended analysis with a pairwise comparison of all clustering results can be found on our website. To handle missing values occurring in some of the datasets due to their sparsity, the distance between features with a missing value is set to the maximal possible distance. Since this principle cannot be applied for OptiComb and PICS, missing entries are imputed here with zero values.

DBLP. In our first experiment we analyze the DBLP data. Authors are represented by vertices and co-authorships by edges. The features consist of 20 keywords extracted from the titles of papers. The keywords are chosen to represent four different fields of research: Data-Mining, Computer Graphics, Artificial Intelligence, and Databases. They include terms like: "classification", "cluster", "graphic", and "human". We used the largest connected component (774 nodes and 1757 edges). The number of clusters is set to 24.

An interesting cluster found by SSCG is a group of 18 scientists from Max Planck Institute, TU Graz and ETH Zurich, all established in the field of computer vision and motion capturing (left ellipse in Figure 7). The relevant dimensions are "motion" and "3d". Another interesting cluster is a set of 20 authors from the field of machine learning and data mining (right ellipse in Figure 7). They are clustered in the dimensions "cluster", "pattern" and "learning".

Table I compares all methods based on internal measures. The methods above the dashed line are the ones considering graph *and* feature information. As expected, SSCG leads to the lowest value for the NSCut. Surprisingly, even though

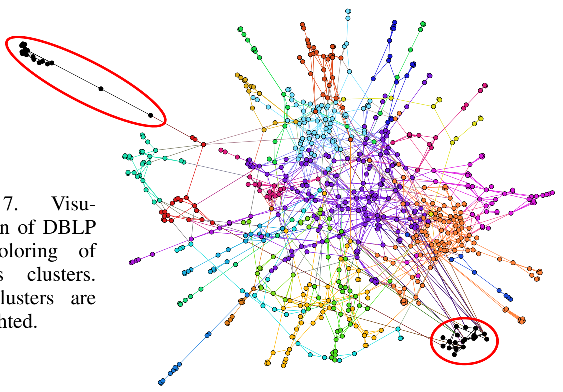

Figure 7. Visualization of DBLP and coloring of SSCG's clusters. Two clusters are highlighted.

Table I: RESULTS ON DBLP

		NMI	TD	NCut	NSCut
using graph and features	**SSCG**	1.000	451.350	4.063	1.774
	CoClus	0.097	456.554	20.789	20.750
	OptiComb	0.083	461.724	21.190	22.177
	SA-Clustering	0.194	444.440	13.860	13.810
	PICS	0.103	460.138	19.234	19.943
only one data type	**SpectVec1**	0.142	432.416	17.008	10.151
	SpectVec2	0.135	436.255	17.195	9.806
	Proclus	0.104	557.686	20.984	18.465
	SpectGraph	0.395	454.823	4.355	5.848

Table II: DFB DATA

		NMI	TD	NCut	NSCut
using graph and features	**SSCG**	1.000	117.448	9.313	9.313
	CoClus	0.200	173.977	12.661	12.445
	OptiComb	0.483	175.953	10.507	10.548
	SA-Clustering	0.395	171.361	11.515	11.500
	PICS	0.223	162.488	12.236	11.966
only one data type	**SpectVec1**	0.184	107.954	13.291	11.581
	SpectVec2	0.209	107.482	13.038	11.098
	Proclus	0.202	21.3495	13.576	12.365
	SpectGraph	0.576	171.646	10.639	10.565

Table III: ARXIV1 DATA

		NMI	TD	NCut	NSCut
using graph and features	**SSCG**	1.000	6.027	1.617	1.420
	CoClus	0.079	5.306	16.623	16.606
	OptiComb	0.050	5.396	17.752	17.744
	SA-Clustering	0.145	9.366	11.900	11.900
	PICS	0.177	5.504	15.154	15.150
only one data type	**SpectVec1**	0.080	5.080	16.733	16.700
	SpectVec2	0.089	5.044	16.118	16.073
	Proclus	0.067	4.081	16.962	16.955
	SpectGraph	0.813	5.413	1.988	1.977

Table IV: ARXIV2 DATA

		NMI	TD	NCut	NSCut
using graph and features	**SSCG**	1.000	18.730	1.546	0.179
	CoClus	0.006	15.437	8.445	8.443
	OptiComb	dnf	dnf	dnf	dnf
	SA-Clustering	dnf	dnf	dnf	dnf
	PICS	0.030	22.383	6.881	6.880
only one data type	**SpectVec1**	dnf	dnf	dnf	dnf
	SpectVec2	0.020	15.408	7.537	7.505
	Proclus	0.008	17.346	8.228	8.056
	SpectGraph	0.145	15.493	2.358	2.340

SSCG does not minimize the (usual) NCut value, its result is better than the one of SpectGraph (which tries to optimize the NCut). Among the methods considering graph and feature information, SSCG obtains the second best TD value, while clearly outperforming these methods w.r.t. the NCut value. As indicated by the low NMI values of the competing methods, SSCG is able to reveal a novel clustering structure.

German soccer premier league. For our next experiment, we extracted the top 100 goal getters from the German soccer premier league. Each node represents a player. Two players are connected if they played in the same soccer club (not necessarily at the same time). As features we chose "number of games", "number of goals", "number of penalty kicks", "average number of goals per game", and "number of soccer clubs". The number of clusters is set to 14.

One interesting cluster is a subset of players from "Borussia Dortmund" being tightly located in the dimensions "number of clubs" (values are spread within a range of 40%) and "number of goals" (range 20%). All the other dimensions are spread across a range of at least 75%. Similarly, a subset of players from "1. FC Nuernberg" are clustered in the dimensions "number of goals" (range 8%), "number of goals per game" (range 18%), and "number of penalty kicks" (range 12%). The remaining dimensions show a spread of at least 35%. None of the competing methods was able to detect such a meaningful clustering structure.

The values of SSCG for the NSCut and the NCut are the overall best (Table II). Among all approaches that use the graph structure, SSCG obtains the best feature compactness (TD). Again, the NMI value indicates that the competing methods detect different results than SSCG.

arXiv. In the arxiv data, papers are represented by nodes, citations by edges, and features denote how often a specific keyword appears in the abstract of the paper. In our first experiment (arxiv1), we used the top 30 keywords and removed nodes showing no keyword, resulting in 856 nodes and 2660 edges. The number of clusters is set to 19.

SSCG found a cluster of 20 papers concerning *quantum gravity*, especially *Lorentzian and Euclidean Quantum Gravity*. The relevant dimensions of this cluster are: "spacetime", "geometry", "gravity", and "integral". Another cluster consists of 16 papers concerning String Theory or more general M-Theory. The papers are about different dimensional branes, dualities and supersymmetry. The relevant dimensions are: "duality", "point", "dimension", and "equation". Here SSCG is especially meaningful since clusters contain well-known (often cited) and also highly topic relevant papers (same keywords) which makes them core papers in their fields.

As shown in Table III, SSCG again shows the best cut values, while maintaining a reasonable distance to the competing approaches regarding TD. Considering the NMI values, most approaches find different clusters than SSCG. Only SpectGraph achieves a similar result (NMI of 0.813).

We also extracted a larger citation-graph having 11,989 nodes, 119,258 edges, and 300 dimensions (arxiv2, Table IV). The results are in line with the observations made for the smaller dataset. While some approaches are not applicable on this data due to their extreme memory usage (larger than 7GB), SSCG can exploit the sparsity of the network which leads to a tractable eigenvalue-decomposition.

Genes. In our next experiment (Table V) we analyzed a gene interaction network (2,900 nodes; 8,264 edges) where genes are additionally enriched by expression values (115 dim.). While all approaches could be applied on this data, SSCG clearly outperforms them w.r.t. the NSCut value, and only SpectGraph was able to realize a similar NCut score. For this data, we observed the largest differences in the clustering results. A pairwise analysis of all clusterings has revealed that none of them can be considered similar, indicating that this dataset is particular challenging to cluster.

Internet Movie Database. The next dataset is an extract of the IMDb. We used movies with at least 200 rankings and an average ranking of at least 6.5 as nodes. Two movies are connected if they share actors or if there exists a reference (e.g. spoofs or follow ups) to each other. As features, we chose all 21 movie genres. To allow good interpretation we focus on movies produced in USA, Canada, UK, or Germany. We used the largest connected component (862 nodes and 4388 edges) and looked for 30 clusters.

SSCG found a cluster of 19 movies including "Evil Dead II", "Poltergeist", and "Predator" based on the relevant dimensions: "Horror", "Mystery", and "Thriller". Another interesting cluster contains 19 movies concerning Jimi Hendrix, Chuck Berry, and U2. All movies are either biographies or movies about music. Conveniently, the relevant dimensions are: "Biography" and "Music". Finally, there is a cluster of 20 romantic comedies containing movies like "10 Items or Less", "Driving Miss Daisy", and "Feast of Love". These three movies are connected through the actor Morgan

Table V: GENE DATA

	NMI	TD	NCut	NSCut
SSCG	1.000	47.939	6.702	5.539
CoClus	0.026	50.462	15.445	15.406
OptiComb	0.017	45.163	16.546	16.135
SA-Clust.	0.014	46.944	16.201	16.107
PICS	0.017	44.627	16.033	15.950
SpectVec1	0.014	47.458	16.995	16.936
SpectVec2	0.018	47.960	16.734	16.110
Proclus	0.013	20.744	17.214	16.631
SpectGraph	0.044	48.089	8.187	8.391

Table VI: IMDB DATA

	NMI	TD	NCut	NSCut
SSCG	1.000	5.963	18.650	0.563
CoClus	0.053	7.812	28.516	28.516
OptiComb	0.125	7.776	28.169	27.924
SA-Clust.	0.182	9.351	22.169	22.169
PICS	0.149	7.768	26.257	26.255
SpectVec1	0.161	7.648	27.813	27.374
SpectVec2	0.168	7.624	27.684	27.217
Proclus	0.166	5.278	27.839	27.486
SpectGraph	0.282	7.816	16.349	16.256

Freeman. “Comedy”, “Romance”, and “Drama” are the relevant dimensions. As shown in Table VI, SSCG obtains the overall best NSCut value. Considering all methods that are using the network structure and the feature information, SSCG also achieves the best values for TD and NCut.

Patents. Finally, we want to show the applicability of SSCG on a large citation network of patents with 100,000 nodes, 188,631 edges, and 5 dimensions. Most of the methods were not applicable on this data. Particularly, from the methods considering graph and feature information, only SSCG and PICS could be applied. The clustering of SSCG showed the following properties (TD: 22534, NCut: 1.04, NSCut: 0.70), which clearly outperforms the result of PICS (TD: 24411, NCut: 10.18, NSCut: 9.94) w.r.t. all measures. This difference of the clustering results is also indicated by a low NMI value of 0.060.

Overall, as shown by all experiments, SSCG is able to detect meaningful clusters on a variety of real world data sets. The competing approaches generate highly different results. The internal evaluation measures show that the clusters of SSCG are very compact in the feature space (low TD) and also well separated in the graph (low cut values).

VI. CONCLUSION

In this work we proposed a spectral clustering method for graphs with feature vectors. The novelty of our approach is the integration of the subspace clustering principle where we tackle the problem of irrelevant features that possibly differ for each cluster. As a consequence the affinity matrix is not given apriori but depends on the partition as well as the determined features and is, thus, part of the learning process. To tackle the fundamental challenge of comparing the clustering structures for different feature subsets, we defined an objective function that is unbiased w.r.t. the number of relevant features. For efficiently approximating our objective, we developed the algorithm SSCG based on spectral clustering. SSCG was applicable on large datasets and the only method simultaneously achieving high clustering qualities in the presence of many irrelevant features. For a variety of real-world datasets SSCG was able to detect meaningful clusters which are very compact in the feature space and simultaneously well separated in the graph.

Acknowledgments. This work has been partly funded by the DFG-664/11 project. Stephan Günnemann has been supported by a fellowship within the postdoc-program of the German Academic Exchange Service (DAAD).

REFERENCES

[1] R. Ge, M. Ester, B. Gao, Z. Hu, B. Bhattacharya, and B. Ben-Moshe, “Joint cluster analysis of attribute data and relationship data,” *TKDD*, vol. 2, no. 2, pp. 1–35, 2008.

[2] H.-P. Kriegel, P. Kröger, and A. Zimek, “Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering,” *TKDD*, vol. 3, no. 1, pp. 1–58, 2009.

[3] Y. Zhou, H. Cheng, and J. X. Yu, “Graph clustering based on structural/attribute similarities,” *PVLDB*, vol. 2, no. 1, pp. 718–729, 2009.

[4] F. Moser, R. Colak, A. Rafiey, and M. Ester, “Mining cohesive patterns from graphs with feature vectors,” in *SDM*, 2009, pp. 593–604.

[5] S. Günnemann, I. Färber, B. Boden, and T. Seidl, “Subspace clustering meets dense subgraph mining: A synthesis of two paradigms,” in *ICDM*, 2010, pp. 845–850.

[6] S. Günnemann, B. Boden, and T. Seidl, “Finding density-based subspace clusters in graphs with feature vectors,” *DMKD*, vol. 25, pp. 243–269, 2012.

[7] U. Von Luxburg, “A tutorial on spectral clustering,” *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[8] F. R. Bach and M. I. Jordan, “Learning spectral clustering, with application to speech separation,” *JMLR*, vol. 7, pp. 1963–2001, 2006.

[9] S. E. Schaeffer, “Graph clustering,” *Computer Science Review*, vol. 1, no. 1, pp. 27–64, 2007.

[10] D. Hanisch, A. Zien, R. Zimmer, and T. Lengauer, “Co-clustering of biological networks and gene expression data,” *Bioinformatics*, vol. 18, pp. 145–154, 2002.

[11] M. Shiga, I. Takigawa, and H. Mamitsuka, “A spectral clustering approach to optimally combining numerical vectors with a modular network,” in *SIGKDD*, 2007, pp. 647–656.

[12] L. Akoglu, H. Tong, B. Meeder, and C. Faloutsos, “Pics: Parameter-free identification of cohesive subgroups in large attributed graphs,” *SDM*, pp. 439–450, 2012.

[13] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park, “Fast algorithms for projected clustering,” in *SIGMOD*, 1999, pp. 61–72.

[14] D. Yan, L. Huang, and M. I. Jordan, “Fast approximate spectral clustering,” in *SIGKDD*, 2009, pp. 907–916.

[15] J. Shi and J. Malik, “Normalized cuts and image segmentation,” *TPAMI*, vol. 22, no. 8, pp. 888–905, 2000.

[16] M. Genton, “Classes of kernels for machine learning: a statistics perspective,” *JMLR*, vol. 2, pp. 299–312, 2002.

[17] M. Maila and J. Shi, “A random walks view of spectral segmentation,” *AI and STATISTICS (AISTATS)*, 2001.

[18] I. Assent, R. Krieger, E. Müller, and T. Seidl, “DUSC: Dimensionality unbiased subspace clustering,” in *ICDM*, 2007.

[19] V. Kann, “On the approximability of NP-complete optimization problems, Royal Inst. of Tech.” 1992.

[20] A. Condon and R. M. Karp, “Algorithms for graph partitioning on the planted partition model,” *Random Struct. Algorithms*, vol. 18, no. 2, pp. 116–140, 2001.

[21] S. Günnemann, I. Färber, E. Müller, I. Assent, and T. Seidl, “External evaluation measures for subspace clustering,” in *CIKM*, 2011, pp. 1363–1372.

[22] N. X. Vinh, J. Epps, and J. Bailey, “Information theoretic measures for clusterings comparison,” *Journal of Machine Learning Research*, vol. 11, pp. 2837–2854, 2010.