Scalable Constrained Clustering: A Generalized Spectral Method

Mihai Cucuringu
U. of California, Los Angeles

Ioannis Koutis
U. of Puerto Rico, Rio Piedras

Sanjay Chawla
Sydney University

ABSTRACT

We present a principled spectral approach to the well-studied constrained clustering problem. It reduces clustering to a generalized eigenvalue problem on Laplacians. The method works in nearly-linear time and provides concrete guarantees for the quality of the clusters, at least for the case of 2-way partitioning. In practice this translates to a very fast implementation that consistently outperforms existing spectral approaches. We support this claim with experiments on various data sets: our approach recovers correct clusters in examples where previous methods fail, and handles data sets with millions of data points - two orders of magnitude larger than before.

1. INTRODUCTION

Clustering with constraints is indisputably a problem of central importance in the data mining community. The extensive literature reports a plethora of approaches and methods, including *spectral* methods that explore various extensions of the *foundational* 'unconstrained' spectral method by Shi and Malik [24], and Ng et al. [21].

While our present work falls in the general class of spectral methods, it enjoys a number of key distinctive features that set it apart from previous works:

a. The method is a natural **generalization**, rather than a mere extension, of the foundational spectral method. It supersedes it by encompassing it as a special case of constrained clustering. The solution is derived from a geometric embedding that results from a spectral relaxation of an optimization problem, exactly in the spirit of [24, 21].

b. The method is in fact a generalization of a **revised** form of the foundational methods, which explicitly incorporates recent theoretical progress by Lee et. al [17]. To the best of our knowledge, the particular embedding that is analyzed in [17] has never been used before in practical settings. Our experiments indicate that it contributes significantly to the quality of our results, and performs better when compared to previously studied variants [29, 20].

c. The method is **fast** by design. Casting the problem as a generalized eigenvalue problem on graph Laplacians enables the use of the recently discovered fast linear system solvers for Laplacians [12]. This yields a nearly-linear time method capable of dealing with much larger data sets.

d. The method comes with a **theoretical guarantee** for the quality of 2-way constrained partitioning, with respect to the underlying discrete optimization problem. The guarantee is a generalization of the classical Cheeger inequality which provides a guarantee for the unconstrained 2-way partitioning. Similar to [24, 21], our work poses a number of theoretical questions. Concretely, the quality of our results suggests the existence of generalizations of 'higher order' Cheeger inequalities shown in [17].

Roadmap. We begin with the problem definition in Section 1.1, where we also introduce and justify a key **design decision**, which is to merge a subset of the constraints with the input graph. In Section 1.2, we proceed to a high-level **schematic overview** of our approach. We also include a discussion of related work that is not intended to be comprehensive but rather to delineate the key differences of our approach. Here, we also make what we believe is an *important point*. Most of the works from the present literature constitute extensions of the 'base' methods in [24, 21]. In certain cases, this base can now be entirely replaced by our method, potentially yielding further improvements, which is why we expect that our method will motivate further research. Section 2 describes the **components** of our method in detail: the discrete optimization problem, its spectral relaxation, the geometric embedding and the final clustering, along with a review of the algorithms that enable near-linear time computation. We also prove that the classical 'unconstrained' spectral clustering is indeed a special case of our framework. Section 3 provides **theoretical justification** for our method. In particular, we describe the concrete bound on the quality of the 2-way partition provided by our method. Our work raises some interesting theoretical questions about its expected performance in the general k-way case. We include the related **conjecture** in Section 3. In Section 4 we report on a comprehensive set of experiments that have been carried out on diverse data sets to validate our proposed approach. We conclude in Section 5 with possible directions for future work. Finally, the appendix section provides more details related to the numerical computation aspects of the Laplacian solver.

1.1 Problem Definition

The constrained clustering problem is specified by three weighted graphs:

1. The *data graph* G_D which, contains a given number of k clusters that we seek to find. Formally, the graph is a triplet $G_D = (V, E_D, w_D)$, with the edge weights w_D being positive real numbers indicating the level of 'affinity' of their endpoints.

2. The *knowledge graphs* G_{ML} and G_{CL}. The two graphs are formally triplets $G_{ML} = (V, E_{ML}, w_{ML})$ and $G_{CL} = (V, E_{CL}, w_{CL})$. Each edge in G_{ML} indicates that its two endpoints should be in the same cluster, and each edge in G_{CL} indicates that its two endpoints should be in different clusters. The weight of an edge indicates the level of belief placed in the corresponding constraint.

We emphasize that prior knowledge does not have to be exact or even self-consistent, and thus the constraints should not be viewed as 'hard' ones. However, to conform with prior literature, we will use the existing terminology of 'must link' (ML) and 'cannot link' (CL) constraints to which G_{ML} and G_{CL} owe their notation respectively.

In the constrained clustering problem the goal is to find k **disjoint** clusters in the data graph. Intuitively, the clusters should result from cutting a small number of edges in the data graph, while simultaneously respecting as much as possible the constraints in the knowledge graphs. Within this framework, one can define various objective functions that attempt to quantify the general goal. None of these functions is a priori the 'right' one, and many of them are arguably natural. An important consideration in picking an objective function is of course the **hardness** of computing satisfactory approximate solutions. Indeed, our choice of functions will be made with an eye towards nearly-linear time computations, an essential constraint in the analysis of very large data.

A design decision: data as ML constraints. One may argue that the data graph is an implicit encoding of ML constraints. Indeed, pairwise affinities between nodes can be viewed as 'local declarations' that these two nodes should be connected rather than disconnected in a clustering. Therefore, the prior knowledge graph G_{ML} can be viewed as a second set of ML constraints, perhaps associated with a different relative level of confidence α. This motivates us to define an extended data graph as

$$\tilde{G}_D[\alpha] = (V, E_D \cup E_{ML}, w_D + \alpha * w_{ML}).$$

The parameter α may be subject to exploration. Different values of α can produce different clusterings that can be evaluated with an independent criterion that can be problem-dependent. In practice, our computations will be very light-weighted, allowing experimentation with varying values of α. In our experiments we consistently take α to be a constant; the results are not sensitive to varying it.

1.2 Spectral Clustering: An Overview

The classical work in [24, 21] casts the k-way partitioning as a discrete optimization problem, which, by its definition, attempts to capture certain properties that a 'nice' cluster is expected to have. The optimization problem is defined solely in terms of the data graph, and its spectral relaxation gives rise to the generalized eigenvalue problem $Lx = \lambda Dx$ where L is the Laplacian matrix (def. in Sec. 2.1) of the input graph, and D is its diagonal. As a next step, k eigenvectors are computed, and after a post-processing step, the nodes of the graph get embedded in the Euclidean space, where the popular k-means algorithm is finally applied to obtain a partition.

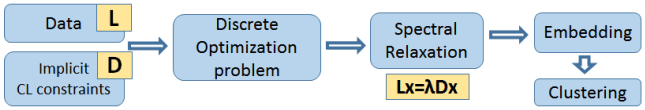


Figure 1: The base spectral method [24, 21].

The algorithmic flow is depicted in Figure 1.2, which takes the non-standard view of the diagonal matrix D as an encoding of *implicit* CL constraints. We will justify this in Section 2.4.

Early works on spectral constrained clustering modified the flow in Figure 1.2 only by changing the Laplacian L in order to incorporate the constraints [9, 15, 19]. As we discussed above, this modification is also part of our approach, but only for the ML constraints.

A different approach is taken in [18], which, as shown in Figure 1.2, modifies the flow after the embedding. This approach is agnostic to the input embedding. In principle, it can be applied onto any embedding including the one returned by our method.

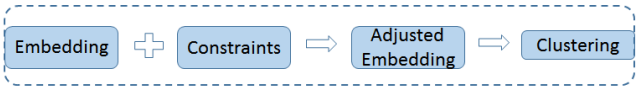


Figure 2: Adjusting the embedding.

A number of other works [33, 5, 10, 31, 30] use the ML and CL constraints to super-impose algebraic constraints onto the spectral relaxation, as shown Figure 1.2.These additional constraints usually give rise to much harder constrained optimization problems. These are solved with algorithms that often skip the embedding component of the algorithmic flow, and have no theoretical guarantees on their runtime. Even the empirical time is certainly not nearly-linear as manifested by the limited data size in the reported experiments, or sometimes by the lack of actual runtime reports.

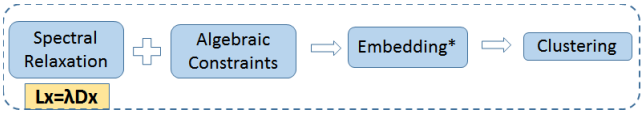


Figure 3: Imposing algebraic constraints.

Adding algebraic constraints can be viewed as a disguised way to re-define the discrete optimization problem. There have also been more explicit attempts to defining a modified discrete optimization problem that incorporates more naturally the constraints [9, 32, 7, 4, 33, 23]. However, none of them deals with both CL constrains and general k-way partitioning.

The work in this article represents a conceptually simple generalization of the original work in [24, 21], and for comparison we show the algorithmic flow in Figure 1.2.

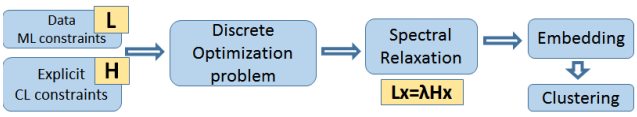


Figure 4: A schematic overview of our approach.

As we discussed above, the ML constraints are merged into the data Laplacian matrix L. The CL constraints (which optionally can include the implicit constraints from Figure 1.2) are now explicitly encoded into a *second Laplacian* H. We then setup a natural discrete optimization problem involving the constraints, and relax it to a generalized eigenvalue problem $Lx = \lambda Hx$. We will carefully describe each of the components in our algorithm, in the next Section.

We wish to close this introductory Section with three somewhat more technical remarks:

R1. The embedding component is of special importance in our approach. Since the introduction of spectral clustering there has been some level of confusion as to whether the eigenvectors of the random walk matrix $D^{-1}L$ or those of the normalized Laplacian $D^{-1/2}LD^{-1/2}$ should be used, and how. This 'confusion' has propagated in various forms into the subsequent literature. Recent work by Lee et al. [17] has settled these questions with the analysis of a slightly revised embedding. In our work, we carefully *generalize* the embedding proposed in [17] using an insight that we derive from another recent work [13]. We have found that using [17] for the underlying 'unconstrained' embedding does yield better results, implying that there is **potential for improvement** in previous works that use the older embeddings.

R2. The work by Wang et al. [30] comes close to our approach in that it reduces the problem into a generalized eigenvalue problem $Lx = \lambda Qx$, where Q encodes both ML and CL constraints. However Q is not always positive definite, and as a result, the problem does not admit a fast algorithm. Conceivably, one can modify the approach in [30] and directly force Q to be positive definite, in particular a *signed Laplacian*. However the algebraic behavior of the signed Laplacian is markedly different and it does not lead to embeddings with good behavior: our experiments have been very disappointing in this direction, in particular for $k > 2$.

R3. A number of algorithms in the literature (e.g. [10, 23]) are iterative, where each iteration provides a slightly better solution to the discrete optimization problem. Naturally, iterative algorithms are expected to be somewhat slower (even when convergence is fast). Our algorithm is a 'one-shot' approximation algorithm: it provides a solution which hopefully is a good approximation to the optimal one. However, it can itself be iterated, for example by adapting the spectral rounding approach [26]. Apart from speed reasons, we deliberately do not discuss iterations because our goal is to describe a simple 'base' approach that can be adjusted, modified or amended by subsequent works.

2. METHOD AND ALGORITHMS

We discuss the components of our methods, following the outline in Figure 1.2. We begin with the definition and a key property of Laplacians.

2.1 Graph Laplacians.

Let $G = (V, E, w)$ be a graph with positive weights. The *Laplacian* L_G of G is defined by $L_G(i,j) = -w_{ij}$ and $L_G(i,i) = \sum_{j \neq i} w_{ij}$. The graph Laplacian satisfies the following basic identity for all vectors x:

$$x^T L_G x \quad = \quad \sum_{i,j} w_{ij}(x_i - x_j)^2.$$

Given a cluster $C \in G$ we define a cluster indicator vector by $x_C(i) = 1$ if $i \in C$ and $x_C(i) = 0$ otherwise. We have:

$$x_C^T L_G x_C \quad = \quad \text{(weight of edges crossing from } C \text{ to } \bar{C}) \quad (1)$$

2.2 The Discrete Optimization Problem

To render our discussion slightly more intuitive, let us assume that the graphs are unweighted. Consider the partition problem in which we seek to split the node set V into S and $V - S$ with the goal of solving the following minimization problem.

$$\phi = \mathbf{min}_S \frac{\#\text{ edges cut in } \tilde{G}_D}{\#\text{ of satisfied CL constraints}}. \quad (2)$$

This is a reasonable objective function for a 2-way cut, because it favors satisfying many CL constraints while violating few ML constraints ('data' and input constraints, as discussed in Section 1.1). This *asymmetry* in the treatment of CL and ML constraints is what enables casting the optimization problems in terms of Laplacians. Concretely, if we let x_C denote the indicator vector for $C \subseteq V$, we have:

$$\phi = \mathbf{min}_C \frac{x_C^T L_{\tilde{G}_D} x_C}{x_C^T L_{G_{CL}} x_C}. \quad (3)$$

The above problem can in principle be generalized to k-way cuts. However, when it comes to partitioning into more clusters it leaves something to be desired. Specifically, it favors the extraction of clusters that are good *on average*, as it pays no attention to individual clusters. With this in mind, we define an **individual measure** of goodness for each cluster C_i among k clusters:

$$\phi_i(\tilde{G}_D, G_{CL}) = \frac{\#\text{ edges leaving } C_i \text{ in } \tilde{G}_D[\alpha]}{\#\text{ satisfied CL constraints incident to } C_i}. \quad (4)$$

where a constraint is said to be incident to C_i if at least one of its endpoints is in C_i. It can be seen that this number of satisfied constraints is equal to the number of edges leaving C_i in G_{CL} We would like then to find clusters $C_1, \ldots, C_k$ that solve the following problem:

$$\Phi_k = \mathbf{min} \max_i \phi_i. \quad (5)$$

While insisting on the worst cluster being good, this definition is flexible in a key way: the clusters do not have to constitute a partitioning of the set of nodes, i.e. their union does not have to be V. Therefore the definition naturally allows and can accommodate for noise or outliers in the data or prior knowledge.

Again, this can be captured in terms of Laplacians: letting C_i denote the indicator vector for cluster i, we have

$$\phi_i(\tilde{G}_D, G_{CL}) = \frac{x_{C_i}^T L_{\tilde{G}_D} x_{C_i}}{x_{C_i}^T L_{G_{CL}} x_{C_i}}.$$

Therefore, solving the minimization problem posed in equation 5 amounts to finding k vectors in $\{0,1\}^n$ with **disjoint** support.

2.3 Spectral Relaxation

In the sequel we will simplify our notation and use G and H to denote $\tilde{G}_D$ and G_{CL} respectively. We have shown in equation 3 that given two graphs G and H we have:

$$\phi(G,H) = \min_{x \in \{0,1\}^n} \frac{x^T L_G x}{x^T L_H x}.$$

A natural step towards approximately solving the problem is to relax it over the reals and find a real vector x which minimizes the following:

$$\lambda(L_G, L_H) = \min_{x \in \mathbb{R}^n} \frac{x^T L_G x}{x^T L_H x}.$$

Note that H does not have to be connected. Since we're looking for a minimum of this ratio, the optimization function automatically avoids vectors that are in the null space of L_H. That means that no restriction needs to be placed on x so that the problem is well defined, other than it can't be the constant vector, assuming without loss of generality that G is connected. It is well understood that the solution vector to this problem is the first non-trivial generalized eigenvector of the problem

$$L_G x = \lambda L_H x \tag{6}$$

and $\lambda(L_G, L_H)$ is the corresponding eigenvalue.

Relaxing the k-way problem. Recall the k-clustering problem we defined in equation 5 and that in order to solve it, it suffices to find k vectors in $\{0,1\}^n$ of **disjoint** support. Imagine that an $n \times k$ matrix X with these vectors as columns is given. Each row of that matrix can be viewed as a point in $\mathbb{R}^k$, and since coordinates of these vectors correspond to graph nodes, each node is mapped into the k-dimensional Euclidean space. Observe that nodes that belong in the same cluster are actually mapped to the same point in $\mathbb{R}^k$. Of course it is hard to compute X, but we can hopefully compute an approximation to it by relaxing the problem over the real vectors. In the relaxed problem, disjointness loosely translates to some type of **orthogonality** and so the goal becomes to find k orthogonal vectors such that the ratios $(x^T L_G x / x^T L_H x)$ are as small as possible. With the appropriate notion of orthogonality the problem is equivalent to computing the k first non-trivial eigenvectors of the problem $L_G x = \lambda L_H x$. These vectors can be used to embed the nodes into $\mathbb{R}^k$ and intuitively the nodes that should be in the same cluster are mapped to nearby points. Then we can use a distance-based separation algorithm like `kmeans` to recover the clusters. This is precisely the idea underlying the original works in [24, 21], with the only difference being in the fact that the diagonal matrix D is used in place of L_H.

2.4 A special case: 'Unconstrained' Spectral Clustering

We claim that the classical spectral relaxation for unconstrained spectral clustering is a special case in our generalized framework.

Before we proceed to a more formal statement and the proof, let us introduce some notation. Given the graph G, let $d_i = \sum_{j \neq i} w_{ij}$ be the weighted degree of vertex i. Let D be the diagonal matrix with $D_{ii} = d_i$.

PROPOSITION 2.1. *Given a connected graph G, there is a graph K such that the generalized eigenvalue problems $L_G x = \lambda D x$ and $L_G x = \lambda L_K x$ have identical non-trivial eigenvectors (and eigenvalues).*

PROOF. Finding the non-trivial generalized eigenvectors for the problem $L_G x = \lambda D x$ can be reduced (via a simple manipulation and similarity transformation) to finding the non-trivial eigenvectors y of the normalized Laplacian

$$D^{-1/2} L_G D^{-1/2} y = \lambda y,$$

and then setting $y = D^{1/2} x$.

Because L_G is a Laplacian, its null space is the constant vector. Therefore, we get that the null space of the normalized Laplacian is the vector whose i^{th} coordinate is $d_i^{1/2}$. The non-trivial eigenvectors y, i.e. the ones that correspond to the non-zero eigenvalue, are orthogonal to the null space of the matrix. So, we have $\sum_i d_i^{1/2} y_i = 0$, which directly implies that $\sum_i d_i x_i = 0$, or more succinctly $d^T x = 0$.

We define K as a complete graph $K = (V, E, w_K)$ with weights given by $w_{ij} = d_i d_j / \sum_i d_i$. It can be verified that

$$L_K = D - \frac{d d^T}{\sum_i d_i}.$$

But then, using $d^T x = 0$ for all vectors x we have

$$L_K x = Dx - \frac{d(d^T x)}{\sum_i d_i} = Dx.$$

That is, under the constraint $d^T x = 0$, the two eigenvalue problems are identical. □

Moreover, the discrete optimization problem underlying the selection of K as a graph of CL constraints has an optimal value which is within a small constant from the optimal of the classical $NCut$ function.

To see this, for a set $S \subseteq V$, let us denote $V(S) = \sum_{i \in S} d_i$. For two disjoint sets $S_1 \subseteq V$ and $S_2 \subseteq V$ we denote by $cut(S_1, S_2)$ the total weight of edges whose endpoints lie in S_1 and S_2. It is not difficult to show that

$$\phi(G,K) = \min_S \frac{cut(S, \bar{S}) vol(V)}{vol(S) vol(\bar{S})}.$$

and

$$NCut(S, \bar{S})/4 \leq \phi(G,K) \leq NCut(S, \bar{S}),$$

where

$$Ncut(S, \bar{S}) = \frac{cut(S, \bar{S})}{vol(S)} + \frac{cut(S, \bar{S})}{vol(\bar{S})}.$$

2.5 The embedding

Let X be the $n \times k$ matrix of the first k generalized eigenvectors for $L_G x = \lambda L_H x$. The embedding is shown in Figure 5.

Input: X, L_H, d
Output: embedding $U \in \mathbb{R}^{n\times k}$, $l \in \mathbb{R}^{n\times 1}$
1: $u \leftarrow 1^n$
2: **for** $i = 1 : k$ **do**
3: $\quad x = X_{:,i}$
4: $\quad x = x - (x^T d/u^T d)u$
5: $\quad x = x/\sqrt{x^T L_H x}$
6: $\quad U_{i,:} = x$
7: **end for**
8: **for** $j = 1 : n$ **do**
9: $\quad l_j = ||U_{j,:}||_2$
10: $\quad U_{j,:} = U_{j,:}/l_j$
11: **end for**

Figure 5: Embedding Computation (based on [17])

We wish to discuss some intuition behind the embedding. Without step 4 and with L_H replaced with the diagonal D, the embedding is exactly the one recently proposed and analyzed in [17]. It is a variant (actually a combination) of the embeddings considered in [24, 21, 29], but the first known to produce clusters with theoretical guarantees that we discuss in Section 3.2. The generalized eigenvalue problem $Lx = \lambda Dx$ can be viewed as a simple eigenvalue problem over a space endowed with the D-inner product: $\langle x, y\rangle_D = x^T Dy$. Step 5 normalizes the eigenvectors to a unit D-norm, i.e. $x^T Dx = 1$. Given this normalization, it is shown in [17] that the rows of U at step 7 (vectors in k-dimensional space) are expected to concentrate in k different *directions*. This justifies Steps 8-10 that normalize these row vectors onto the k-dimensional sphere, in order to concentrate them in a *spatial* sense. Then a geometric partitioning algorithm can be applied.

From a technical point of view, working with L_H instead of D makes almost no difference. L_H is a positive definite matrix. It can be rank-deficient, but the eigenvectors avoid the null space of L_H, by definition. Thus the geometric intuition about U remains the same if we syntactically replace D by L_H. Of course, there is a subtlety: L_G and L_H share the constant vector in their null spaces. This means that if x is an eigenvector, then for all c the vector $x + c\mathbf{1}^n$ is also an eigenvector with the same eigenvalue. Among all such possible eigenvectors we pick one representative: in Step 4 we pick c such that $x + c\mathbf{1}^n$ is orthogonal to d. The intuition for this is derived from [13]; this choice is what makes possible the analysis of a theoretical guarantee for a 2-way cut. The choice of d (the vector of weighted degrees in G), also reflects mathematical evidence that the quality of the embedding actually depends only on properties of graph G, not both G and H. We will further discuss this in Section 3.3.

2.6 Clustering

Given the embedding matrix embedding U, the clustering algorithm invokes `kmeans`(U), which returns a k-partitioning. The partitioning can be refined optionally into a k-clustering by performing a so-called Cheeger sweep [3] among the nodes of each component, independently for each component. To perform the Cheeger sweep, the nodes should be sorted according to the values of the corresponding coordinates in the vector l returned by the embedding algorithm given in 5. We won't use this option in our experiments. For the special case when $k = 2$, we only need to find the second eigenvector, sort it and then perform a Cheeger sweep.

2.7 Computing Eigenvectors

The computation of the k generalized eigenvectors for $L_G x = \lambda L_H x$ is the most time-consuming part of the entire process. It takes up to $O(km \log^2 m)$ time, where m is the number of edges in G and H. Empirically, the running time is $O(km \log m)$. Algorithms for computing eigenvectors of Laplacians are well-understood. They depend crucially on fast linear system solvers for Laplacians [11, 14]. We discuss some details in the Appendix section.

3. THEORETICAL JUSTIFICATION AND OPEN QUESTIONS

3.1 A generalized Cheeger inequality

Spectral partitioning methods were developed on the basis of what is known as the *Cheeger inequality* [3]. Specifically, if y is any vector such that $y^T d = 0$ then we have

$$\frac{y^T L_G y}{y^T D y} \geq \frac{(\phi(G))^2}{2}. \tag{7}$$

This means that the sparsest cut in the graph cannot be larger than the square root of the ratio of quadratic in the left-hand side. The proof is actually constructive, as it finds a cut at least as good as the guarantee, by using the so-called *Cheeger sweep*: if $S_i \subseteq V$ is the set of nodes corresponding to the i smallest entries of y, then there is some i such that

$$\lambda(L_G, L_K) \leq \frac{cut_G(S_i, \bar{S}_i)}{cut_K(S_i, \bar{S}_i)} \leq 2\sqrt{\frac{y^T L_G y}{y^T D y}}.$$

Recall that $\lambda(L_G, L_K)$ denotes the smallest non-trivial eigenvalue of $L_G x = \lambda L_K x$. One would like the right-hand side of the above inequality to be as small as possible, which in general can be achieved by computing an 'approximate' eigenvector, i.e. a vector y which minimizes this ratio within a constant of 2.

But is something similar possible for arbitrary pairs of graphs G and H? It was recently shown that a generalization of it *is* possible [13]: one can use any vector y such that $y^T d = 0$ to find (via a Cheeger sweep) a set S satisfying

$$\lambda(L_G, L_K) \leq \frac{cut_G(S, \bar{S})}{cut_H(S, \bar{S})} \leq 4\frac{y^T L_G y}{y^T L_H y} \cdot \frac{1}{\phi(G, K)}. \tag{8}$$

This generalized Cheeger inequality shows that in the 2-way case our algorithm returns a cut which can be at most $\phi(G, K)$ away from the optimal cut.

3.2 Higher-order Cheeger inequalities

The applied success of spectral clustering [24, 21] led to a conjecture for the existence of Cheeger-like inequalities involving the higher eigenvalues of $Lx = \lambda Dx$. The conjecture was **settled** in the positive in the relatively recent work of Lee et. al [17]. Specifically it was shown that from the embedding of the generalized eigenvectors, one can recover k clusters $C_1, \ldots, C_k$, such that for all i:

$$\lambda_k(L_G, D) \leq \phi_i(G, K) \leq g(k)\sqrt{\lambda_k(L_G, D)}$$

where k is the k^{th} eigenvalue of $L_G x = \lambda Dx$, and $g(k)$ is a small function of k; since in our case k will be a small constant the reader can think of $g(k)$ as a constant. We

conjecture that a graph-dependent, high-order generalized analogue of 8 exists and that it is possible to compute clusters satisfying:

$$\phi_i(G,H) \leq g'(k)\frac{\lambda_k(L_G,L_H)}{\Phi_k(G,K)},$$

for some small function $g'(k)$. We believe that our experiments provide supporting evidence. The fact that the graph-dependent Cheeger inequalities in [16] have higher order analogues (for the pair (G,K)) constitutes additional mathematical evidence.

3.3 Graph dependent Cheeger inequalities

Inequality 8 is interesting in an additional subtle way: the approximation quality depends *only* on G. There have been recent results that provide graph-dependent Cheeger inequalities [16] for the pair of graphs (G,K). These show that the standard Cheeger inequality is **pessimistic** for large classes of natural graphs (e.g. graphs with a small number of small eigenvalues) and thus partially explain its practical success. Inequality 8 can be viewed as the basic graph-dependent Cheeger inequality for the generalized problem. It is actually possible to derive other variants of it, similar to the ones claimed in 8 but this is beyond the scope of this article.

A number of recent works (e.g. [1, 2]) observe that the output of plain spectral clustering improves when the input graph/matrix is perturbed. We speculate that this is also a by-product of our design decision to merge the ML constraints with the input graph. In particular, ML constraints often connect vertices that are not close in the data graph. This in effect alters the graph significantly, in a spectral sense, without however dramatically changing its clustering (since ML constraints represent the same 'ground truth' where data comes from). This can potentially create situations where the Cheeger inequality is indeed pessimistic and the spectral relaxation provides outputs much better than expected in the general case.

4. EXPERIMENTS

In this section, we sample some of our experimental results. We compare our method (which we denote by **Fast-GE**) mostly against the state-of-the-art constrained spectral clustering algorithm (denoted by **CSP**) proposed in [30], which has been already shown to outperform other approaches for constrained spectral clustering. For very small data sets we can replace the iterative methods in our code by exact algorithms and actually gain in speed. We report the running time of this exact version under the name **GE**. We also include a few comparisons with the classical (unconstrained) spectral algorithm by Shi and Malik [24].

We run our experiments on an inexpensive mobile 2.4 GHz Intel Core i7 processor with 8 GB 1600 MHz DDR3 memory. We wish to emphasize that the reported running times are only *indicative* of what is possible, as our code is far from optimized. With appropriate optimizations we expect a 2-3x speed-up. The algorithms are also highly parallelizable.

4.1 Synthetic Data Sets

The adjacency graphs underlying the three synthetic data sets we consider are constructed as follows. We say that a graph G is generated from the ensemble $NoisyKnn(n,k_g,l_g)$ with parameters n, k_g and l_g if G of size n is the union of two (non-necessarily disjoint) graphs H_1 and H_2 each on the same set of n vertices $G = H_1 \cup H_2$, where H_1 is a k-nearest-neighbor (knn) graph with each node connected to its k_g nearest neighbors, and H_2 is an Erdős-Rényi graph where each edge appears independently with probability $\frac{l_g}{n}$. One may interpret the parameter l_g as the noise level in the data, since the larger l_g the more random edges are wired across the different clusters, thus rendering the problem more difficult to solve. In other words, the *planted* clusters are harder to detect when there is a large amount of noise in the data, obscuring the separation of the clusters.

Since in the above synthetic data sets, the ground truth partition is available, we measure the accuracy of the methods by the popular Rand Index [22]. The Rand Index indicates how well the resulting partition matches the ground truth partition; a value closer to 1 indicates an almost perfect recovery, while a value closer to 0 indicates an almost random assignment of the nodes into clusters. We also remark that the labeled nodes, that result in the aforementioned must-link and cannot-link constraints, are chosen uniformly at random from the set of n nodes.

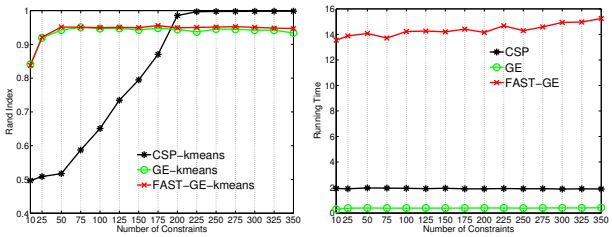
Four Moons. As a first synthetic data set, we consider the Four-Moon example where the underlying graph G is generated from the ensembles $NoisyKnn(n=500,k_g=30,l_g=3)$ (Figure 6) and $NoisyKnn(n=1500,k_g=30,l_g=15$ (Figure 7). As illustrated by Figures 6 (a) and 7 (a), we point out that the accuracy of the CSP method diminishes significantly on this data set with $k=4$ clusters compared to the case $k=2$ (for which we omit the experiments due to space considerations). Furthermore, for the noisier ensemble $NoisyKnn(n=1500,k_g=30,l_g=15)$, the CSP method is unable to return any meaningful results, as shown by Figure 7(c). Note that **GE** and **CSP** are faster than **Fast-GE** for smaller graph with $n=500$ nodes, but significantly slower on the larger graph of size $n=1500$.

PACM. Our second synthetic example is the $PACM$ graph, formed by a cloud of $n=426$ points in the shape of letters $\{P,A,C,M\}$, whose topology renders the segmentation particularly challenging. We again consider two ensembles $NoisyKnn(n=436,k_g=30,l_g=3)$ and $NoisyKnn(n=436,k_g=30,l_g=15)$, and point out that the latter one is a **very noisy** example since, on average, each node has half of its incident edges randomly wired throughout the graph while the remaining edges connect to nodes in its own cluster. Our approach returns superior results when compared to the **CSP** method, except for a few instances from the second ensemble, when the number of constrains is at least 250, thus accounting for more than 60% of the vertices. However, in most practical applications, the number of constraints available to the user is usually much smaller than 60%. We point out that for the larger and noisier graph, when the number of constrains is less then 250, we produce significantly more accurate results.

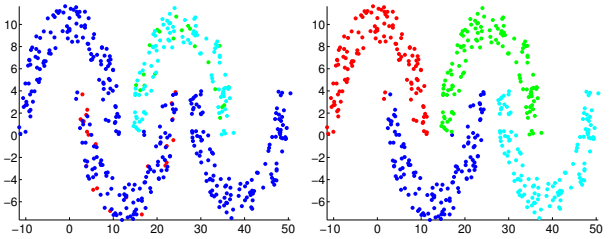
4.2 Real Data

In terms of real data, we consider two very different applications. Our first application is to segmentation of real images, where the underlying grid graph is given by the affinity matrix of the image, computed using the RBF kernel, based on the grayscale values of the [8]. The second data set comes from the social networks literature, and represents friendship Facebook networks in American colleges.

Santorini. In Figure 10 we test our proposed method on

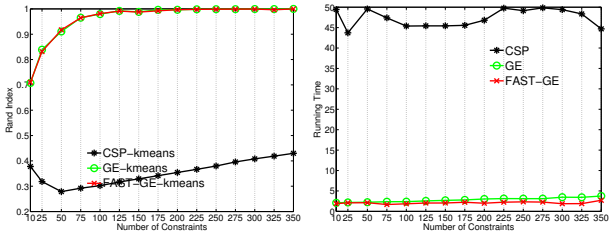


(a) Accuracy (b) Running times

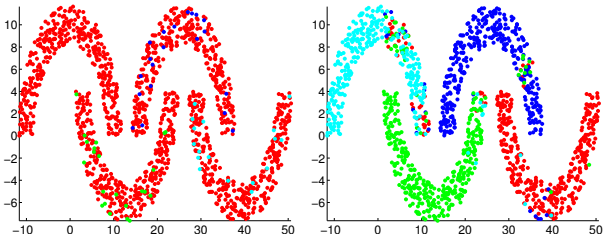


(c) CSP, $C = 75$ (d) FAST-GE, $C = 75$

Figure 6: Accuracy and running times for the Four-Moons data set, with $n = 500$ nodes, and an underlying graph is given by the model NoisyKnn($k = 30, l = 3$), as we vary the number of constraints. We average the results over 100 experiments.

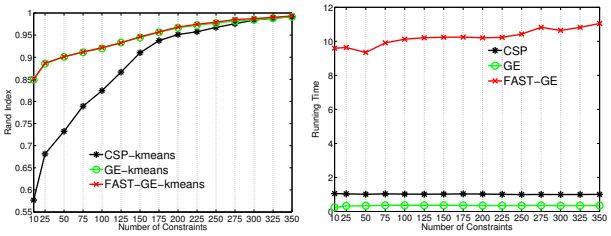


(a) Accuracy (b) Running times

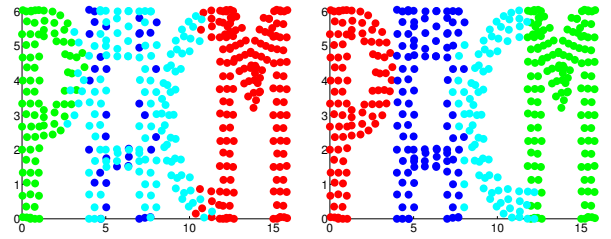


(c) CSP, $C = 75$ (d) FAST-GE, $C = 75$

Figure 7: Accuracy and running times for the Four-Moons data set, with $n = 1500$ nodes, and an underlying graph given by the model NoisyKnn($k = 30, l = 15$), as we vary the number of constraints. We average the results over 100 experiments.

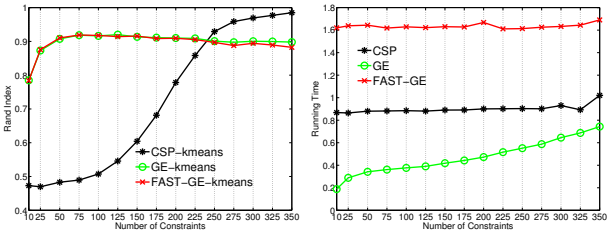


(a) Accuracy (b) Running times

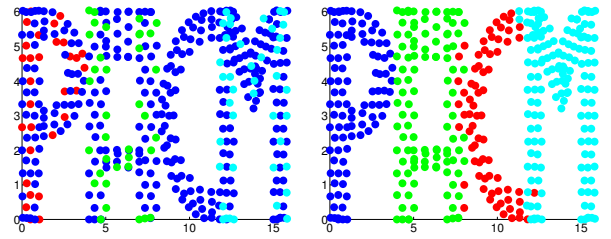


(c) CSP, $C = 125$ (d) FAST-GE, $C = 125$

Figure 8: Accuracy (a) and running times (b) for the PACM data set, with G given by the ensemble NoisyKnn($n = 426$, $k = 30, l = 3$), for different number of constraints. We average the results over 10 experiments. Segmentation for a random instance produced by CSP (c) and FAST-GE (d).



(a) Accuracy (b) Running times



(c) CSP, $C = 125$ (d) FAST-GE, $C = 125$

Figure 9: Accuracy and running times for the PACM data set, where G is given by the ensemble NoisyKnn($n = 426$, $k = 30, l = 15$), as we vary the number of constraints. We average the results over 10 experiments.

the very large *Santorini* image, of size 628×419, with a total of over a quarter million pixels. The difficulty of the segmentation is amplified one one hand by the rather large number of clusters, and on the other hand by the less clear boundaries between some of the pairs of clusters. Although the image is somewhat difficult to reconstruct, our approach successfully recovers the main regions, with few errors, in just **130 seconds**. Computing clusterings in data of this size is infeasible for CSP.

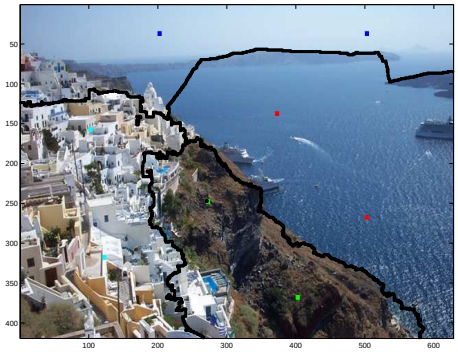


Figure 10: *Santorini*

Patras. Figure 11 shows the segmentation of an image with over 44K pixels and 5 clusters, which our method is able to detect satisfactorily, in under **35 seconds**. In Figure 13 we consider a sub-image of the one used in Figure 11 consisting of only 1886 pixels, in order to be able to compare with CSP and plain unconstrained spectral clustering. CSP takes **139 seconds** to perform the computation, while both **Fast-GE** and unconstrained clustering compute a partition in around **1 second**. The outcome of both CSP and the unconstrained clustering algorithms are clearly not satisfactory.

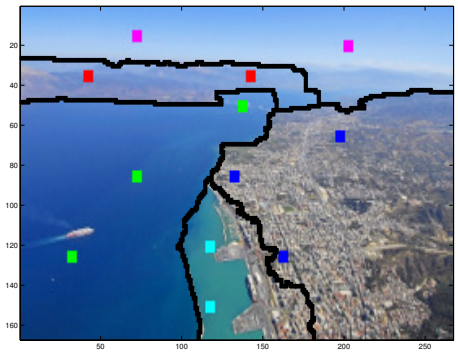


Figure 11: *Patras*

Soccer. Finally, in Figure 12 we consider one last *Soccer* image, with over 1.1 million pixels for which we compute the resulting partitioning into $k = 5$ clusters, using the **Fast-GE** method, in under **12 minutes**. Note that while k-means clustering hinders some of the details in the image, the individual eigenvectors are able to capture more granular details, such as the soccer ball for example.

4.3 Friendship Networks

The authors of [27, 28] study the structure of Facebook friendship networks at one hundred American colleges and universities at a single point in time (2005) and investigate the community structure at each institution, as well as the impact and correlation of various self-identified user characteristics (such as residence, class year, major, and high school) with the identified network communities.

While at many institutions, the community structures are organized almost exclusively according to class year, as pointed out in [27], California Institute of Technology (Caltech) is well-known to be organized almost exclusively according to its undergraduate *House* system (dormitory residence), which is very well reflected in the identified communities. To this end, it is a natural assumption to consider the dormitory affiliation as the ground truth clustering, and aim to recover this underlying structure from the available friendship graph and any available constraints. We add constraints to the clustering problem by sampling uniformly at random nodes in the graph, and the resulting pairwise constraints are generated depending on whether the two nodes belong to the same cluster or no. In order for us to be able to compare to the computationally expensive **CSP** method, we consider several small-sized schools, in particular Caltech (with $n = 590$ nodes, average degree $\bar{d} = 43$, and $k = 8$ clusters), Simmons College ($n = 850$, $\bar{d} = 36$, $k = 10$) and Haverford College ($n = 1025$, $\bar{d} = 72$, $k = 15$).

5. FUTURE WORK

There are several ways our work can be improved or extended: (i) Higher quality results may be obtainable via more sophisticated ways of placing and weighting constraints. (ii) Apart from various optimizations which can speed-up our current implementation at least by a 2x factor, it may be in fact possible to implement a multi-level algorithm for finding the eigenvectors in order to eliminate the $\log n$ factor induced by their computation; in practice this could result in a 10x speed up for the larger data sets in our experiments. (iii) Figure 12 reported indicates that eigenvectors can potentially provide more information. Designing algorithms that can exploit this information is an interesting problem.

6. REFERENCES

[1] K. Chaudhuri, F. C. Graham, and A. Tsiatas. Spectral clustering of graphs with general degrees in the extended planted partition model. In S. Mannor, N. Srebro, and R. C. Williamson, editors, *COLT 2012 - The 25th Annual Conference on Learning Theory, June 25-27, 2012, Edinburgh, Scotland*, volume 23 of *JMLR Proceedings*, pages 35.1–35.23. JMLR.org, 2012.

[2] A. Chen, A. A. Amini, P. J. Bickel, and E. Levina. Fitting community models to large sparse networks. *CoRR*, abs/1207.2340, 2012.

[3] F. Chung. *Spectral Graph Theory*, volume 92 of *Regional Conference Series in Mathematics*. American Mathematical Society, 1997.

[4] T. Coleman, J. Saunderson, and A. Wirth. Spectral clustering with inconsistent advice. In W. W. Cohen, A. McCallum, and S. T. Roweis, editors, *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, volume 307 of *ACM International Conference Proceeding Series*, pages 152–159. ACM, 2008.

[5] A. P. Eriksson, C. Olsson, and F. Kahl. Normalized cuts revisited: A reformulation for segmentation with linear grouping constraints. *Journal of Mathematical Imaging and Vision*, 39(1):45–61, 2011.

[6] G. Golub and C. V. Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, 3d edition, 1996.

[7] L. Grady. Multilabel random walker image segmentation using prior models. In *2005 IEEE Computer Society*

Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA, pages 763–770. IEEE Computer Society, 2005.

[8] L. Grady and E. L. Schwartz. The graph analysis toolbox: Image processing on arbitrary graphs. Technical report, Boston University, Boston, MA, 2003.

[9] S. D. Kamvar, D. Klein, and C. D. Manning. Spectral learning. In *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 561–566, 2003.

[10] J. Kawale and D. Boley. Constrained spectral clustering using l1 regularization. In *SDM*, pages 103–111, 2013.

[11] I. Koutis, G. L. Miller, and R. Peng. A nearly $m \log n$ solver for SDD linear systems. In *FOCS '11: Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science*, 2011.

[12] I. Koutis, G. L. Miller, and R. Peng. A fast solver for a class of linear systems. *Commun. ACM*, 55(10):99–107, Oct. 2012.

[13] I. Koutis, G. L. Miller, and R. Peng. A Generalized Cheeger Inequality. *CoRR*, abs/1412.6075, 2014.

[14] I. Koutis, G. L. Miller, and D. Tolliver. Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing. In *International Symposium of Visual Computing*, pages 1067–1078, 2009.

[15] B. Kulis, S. Basu, I. Dhillon, and R. Mooney. Semi-supervised graph clustering: A kernel approach. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 457–464, New York, NY, USA, 2005. ACM.

[16] T. C. Kwok, L. C. Lau, Y. T. Lee, S. O. Gharan, and L. Trevisan. Improved Cheeger's Inequality: Analysis of Spectral Partitioning Algorithms through Higher Order Spectral Gap. *CoRR*, abs/1301.5584, 2013.

[17] J. R. Lee, S. Oveis Gharan, and L. Trevisan. Multi-way spectral partitioning and higher-order cheeger inequalities. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 1117–1130, New York, NY, USA, 2012. ACM.

[18] Z. Li, J. Liu, and X. Tang. Constrained clustering via spectral regularization. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 421–428, 2009.

[19] Z. Lu and M. Á. Carreira-Perpiñán. Constrained spectral clustering through affinity propagation. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA*. IEEE Computer Society, 2008.

[20] U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[21] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pages 849–856, 2001.

[22] W. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[23] S. S. Rangapuram and M. H. 0001. Constrained 1-spectral clustering. In *AISTATS*, pages 1143–1151, 2012.

[24] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.

[25] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *CoRR*, abs/cs/0607105, 2006.

[26] D. Tolliver and G. L. Miller. Graph partitioning by spectral rounding: Applications in image segmentation and clustering. In *CVPR (1)*, pages 1053–1060, 2006.

[27] A. L. Traud, E. D. Kelsic, P. J. Mucha, and M. A. Porter. Comparing Community Structure to Characteristics in Online Collegiate Social Networks. *SIAM Review*, 53(3):526–543, Aug. 2011.

[28] A. L. Traud, P. J. Mucha, and M. A. Porter. Social structure of Facebook networks. *Physica A: Statistical Mechanics and its Applications*, 2012.

[29] D. Verma and M. Meila. A comparison of spectral clustering algorithms. Technical report, 2003.

[30] X. Wang, B. Qian, and I. Davidson. On constrained spectral clustering and its applications. *Data Min. Knowl. Discov.*, 28(1):1–30, 2014.

[31] L. Xu, W. Li, and D. Schuurmans. Fast normalized cut with linear constraints. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 2866–2873, 2009.

[32] Q. Xu, M. desJardins, and K. Wagstaff. Active constrained clustering by examining spectral eigenvectors. In *Discovery Science, 8th International Conference, DS 2005, Singapore, October 8-11, 2005, Proceedings*, pages 294–307, 2005.

[33] S. X. Yu and J. Shi. Segmentation given partial grouping constraints. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):173–183, 2004.

Appendix

Computing eigenvectors

We compute approximations to the first k eigenvectors via a simple inverse power method [6], i.e. the power method applied to the 'inverse' matrix $A^{-1} = L_G^+ L_H$. Specifically, the algorithms maintains a matrix of current approximate eigenvectors Y. Each iteration computes $Y' = A^{-1}Y$ and then re-assigns to Y an L_H-orthonormal basis for Y' (applying Gram-Schmidt on the columns of Y').This iteration converges in $O(\log n)$ iterations to sufficiently good approximations of the largest eigenvectors of $L_G^+ L_H$. These are identical to the first k non-trivial eigenvectors of $L_G x = \lambda L_H x$. There are iterative methods that are somewhat faster in practice, but we prefer this simple approach because of its provable properties. The power method converges to the largest eigenvectors, and in principle we could obtain them by applying to the matrix $(I - A)$. The fact that we work with the inverse matrix A^{-1} is crucial for fast convergence (see [25] for an intuition).

Solving Laplacian Linear Systems

Multiplications with L_G^+ in the routine computing eigenvectors are actually implemented as linear system solves. That's because $y = L_G^+ x$ is the solution to $L_G y = x$. Recall that L_G is a Laplacian. There are now very fast Laplacian solvers that run in $O(m \log m)$ time where m is the number of non-zeros in A [11]. In practice we use the `CMG` linear system solver which empirically runs in $O(m)$ time for sparse graphs [14][1]. These fast solvers are iterative; they return approximate solutions that can be made arbitrarily good with a sufficient number of iterations. However it is well understood [25], that in order to compute sufficiently good approximations to the eigenvectors, the linear systems need only be solved to a fixed constant accuracy. Thus and a constant number of solver iterations suffices for each linear system solve.

[1]The solver can be obtained from:
`http://www.cs.cmu.edu/~jkoutis/cmg.html`

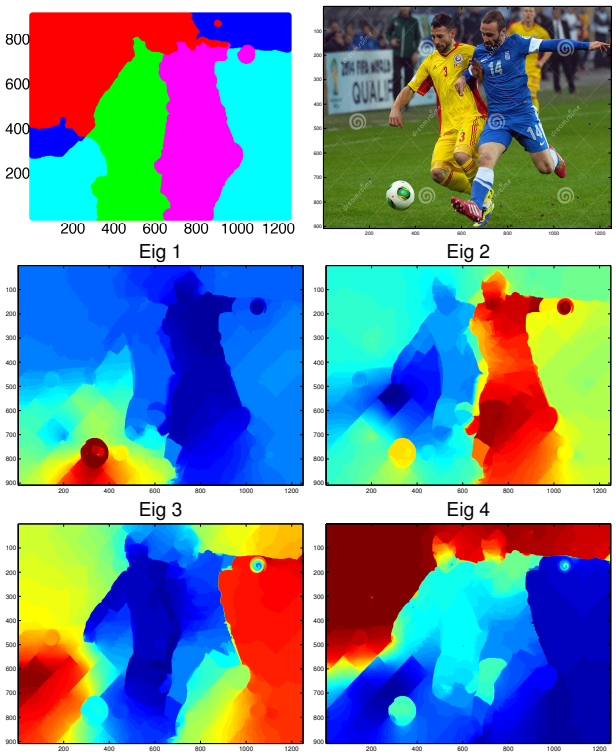


Figure 12: Segmentation and heatmaps of the top four eigenvectors, for the *Soccer* image with over 1.1 million pixels, using our proposed FAST-GE method, in under 12 minutes.

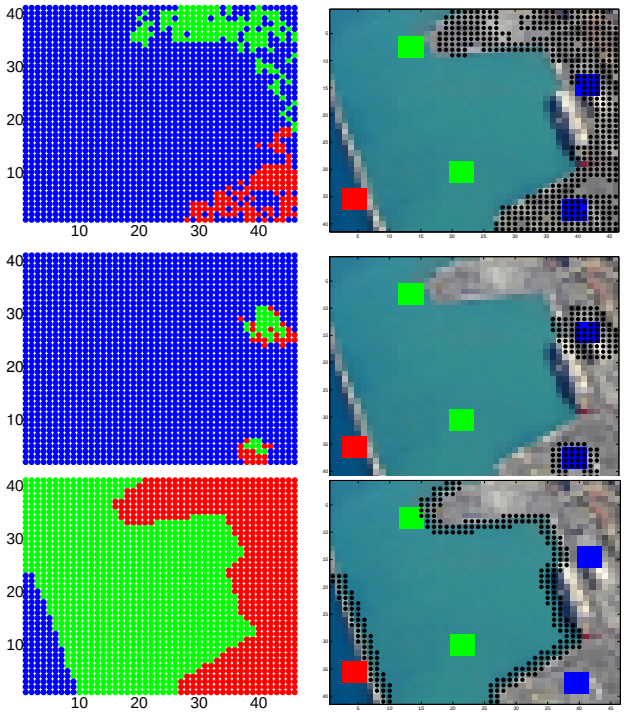
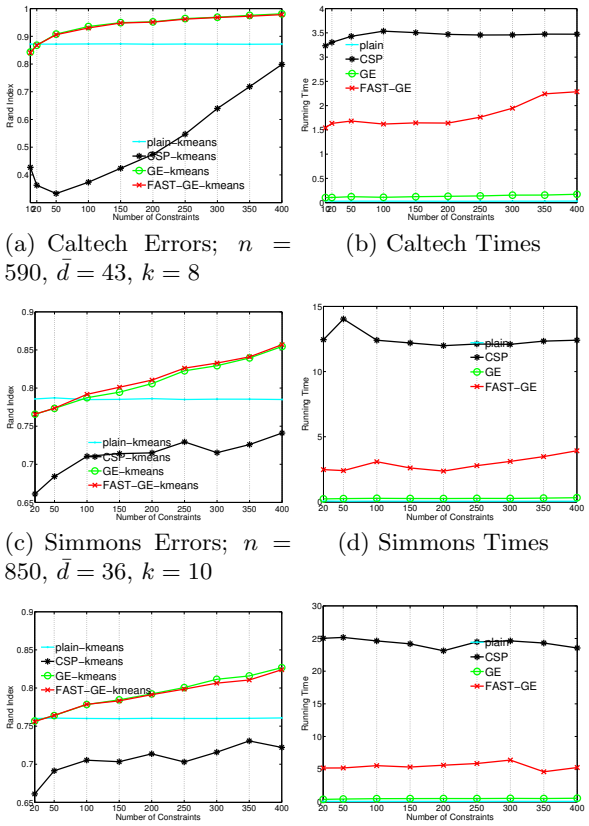


Figure 13: A comparison of unconstrained spectral clustering (row 1), CSP (row 2) and FAST-GE (row 3) for a sub-image of the *Patras* image, with three clusters. The right column overlays the boundary on the actual image.



(a) Caltech Errors; $n = 590$, $\bar{d} = 43$, $k = 8$

(b) Caltech Times

(c) Simmons Errors; $n = 850$, $\bar{d} = 36$, $k = 10$

(d) Simmons Times

(e) Haverford Errors; $n = 1025$, $\bar{d} = 72$, $k = 15$

(f) Haverford Times

Figure 14: *Facebook networks*