J Intell Inf Syst (2015) 44:289–308
DOI 10.1007/s10844-013-0285-0

A scalable approach to spectral clustering with SDD solvers

Nguyen Lu Dang Khoa · Sanjay Chawla

Received: 31 January 2013 / Revised: 22 August 2013 / Accepted: 4 October 2013 /
Published online: 24 October 2013

Abstract The promise of spectral clustering is that it can help detect complex shapes and intrinsic manifold structure in large and high dimensional spaces. The price for this promise is the expensive computational cost for computing the eigen-decomposition of the graph Laplacian matrix—so far a necessary subroutine for spectral clustering. In this paper we bypass the eigen-decomposition of the original Laplacian matrix by leveraging the recently introduced near-linear time solver for symmetric diagonally dominant (SDD) linear systems and random projection. Experiments on several synthetic and real datasets show that the proposed approach has better clustering quality and is faster than the state-of-the-art approximate spectral clustering methods.

Keywords Spectral clustering · Resistance distance · SDD solver · Random projection

1 Introduction

A fast SDD solver (Spielman and Teng 2004, 2006; Koutis et al. 2010, 2011) for linear systems is a new class of near-linear time methods for solving a system of equations $Ax = b$ when A is a symmetric diagonally dominant matrix. It was first proposed in a seminal paper by Spielman and Teng (2004). The Laplacian matrix L of a graph, defined as $L = D - A$, where A is the adjacency matrix and D is the diagonal matrix consisting of degree weights is a SDD matrix. The advantage of the SDD solver is that

N. L. D. Khoa (✉)
National ICT Australia (NICTA), Sydney, Australia
e-mail: khoa.nguyen@nicta.com.au

S. Chawla
School of IT, University of Sydney, Sydney, Australia
e-mail: sanjay.chawla@sydney.edu.au

if a problem can be reduced to computing $Lx = b$ then we can obtain x in near-linear time (in the number of non-zero entries of L).

Now consider the following function on a graph G of n nodes. If i and j are two nodes of G then define:

$$r_{ij} = (e_i - e_j)^T L^+ (e_i - e_j), \tag{1}$$

where L^+ is the pseudo-inverse of L and e_i is the n dimensional column vector with a 1 at location i and zero elsewhere.

r_{ij} is known as the resistance distance in electrical networks (Doyle and Snell 1984). It is a proper metric and is effectively the Euclidean distance in the space spanned by the eigenvectors of the Laplacian L (Fouss and Renders 2007). Therefore k-means clustering using r_{ij} as a distance function is spectral clustering.

With the SDD solver we can solve the system $Lz = e_i - e_j$ in near-linear time. Then $r_{ij} = (e_i - e_j)^T z$ can be also computed in near-linear time. This demonstrates a naive way to use the SDD solver for spectral clustering without direct eigen-decomposition. Given the complexity of k-means as $O(nkd)$ where d is the cost of the distance function, such a naive method is still faster than the $O(n^3)$ eigen-decomposition of the Laplacian.

As shown in Fig. 1, the bottleneck of spectral clustering is the creation of the embedding which involves the eigen-decomposition of L. Therefore, most of the approaches to approximate spectral clustering try to accelerate the embedding step. Although we can make use the sparsity of L in sparse graph by computing a few smallest eigenvectors of L using Lanczos method, the method is still hard to converge and thus is inefficient to use in large graphs (Mavroeidis 2010).

Recent approaches approximate the embedding indirectly by selecting a representative sample and creating the embedding based on the eigen-decomposition in the sample (Fowlkes et al. 2004; Wang et al. 2009; Yan et al. 2009; Chen and Cai 2011). Fowlkes et al. (2004) used traditional Nyström method to solve the eigensystem solution for the representatives which were sampled randomly and then extrapolated the solution for the whole dataset. Yan et al. (2009) performed spectral clustering on a small set of data centers chosen by k-means or a random projection tree. Then all data points were assigned to clusters corresponding to its centers in the center selection step. A recent work in Chen and Cai (2011) used the idea of sparse coding to approximate the affinity matrix based on a number of data representatives so that they can compute the eigensystem very efficiently.

Although the samples or representatives are chosen uniformly at random or by using a different mechanism, it may not completely represent the whole dataset and thus may not correctly capture the cluster geometry structures. Moreover, approximate methods working directly with feature data cannot be used in graph data which are popularly available such as social networks, web graphs, and recommender systems.

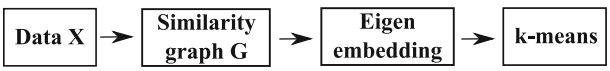


Fig. 1 A typical spectral clustering workflow. Most approaches to speed up spectral clustering use sampling to reduce the cost of eigen-decomposition. Our proposed approach directly reduces the problem with the SDD solver and random projection

We propose an approach to accelerate spectral clustering by directly approximating the eigen embedding via resistance distance without using sampling. It makes use of the SDD solver as described above and random projection to significantly reduce the computational cost while still well maintaining the clustering accuracy. The contributions of this paper are as follows:

- We reformulate the spectral clustering problem using the resistance distance embedding, and make use of the SDD solver and random projection to quickly find the embedding. The method does not use sampling and direct eigen-decomposition like other methods. It can also work directly with graph data. To the best of our knowledge, this is the first attempt to use the SDD solver and random projection to approximate spectral clustering.
- We show the weakness of sampling-based approximate approaches and the strength of the proposed approach in terms of accuracy and performance in several synthetic and real datasets. It is more accurate and faster than the state-of-the-art approximate spectral clustering methods.
- We use random projection to quickly create the k-nearest neighbor graph for spectral clustering in high dimensional data. The experiments in real datasets show that the method can reduce the computational time significantly while still maintaining the clustering accuracy.

The remainder of the paper is organized as follows. Section 2 gives the background on spectral clustering, random projection and the SDD solver. Section 3 describes efforts to approximate spectral clustering to reduce the computational time. In Section 4, we present a method to approximate spectral clustering without direct eigen-decomposition. In Section 5, we evaluate our approach using experiments on synthetic and real datasets. We conclude in Section 6 with a summary and a direction for future research.

2 Background

2.1 Spectral clustering

Given a dataset $X \in \mathbb{R}^{n \times d}$ with n data points, we define an undirected and weighted graph G. Let $A = w_{ij}(1 \leq i, j \leq n)$ be the affinity matrix of G, D be the diagonal degree matrix, and $L = D - A$ be the Laplacian matrix.

Spectral clustering assigns data points in X to k clusters. The details are described in Algorithm 1 (Luxburg 2007).

Algorithm 1 Spectral Clustering

Input: Data matrix $X \in \mathbb{R}^{n \times d}$, number of clusters k
Output: Cluster membership for each data point

1: Construct a similarity graph G from X and compute its Laplacian matrix L.
2: Compute the first k eigenvectors of L.
3: Let $U \in \mathbb{R}^{n \times k}$ be the eigenspace containing these k vectors as columns and each row of U corresponds to a data point in X.
4: Cluster the points in U using k-means clustering.

There are three typical similarity graphs: the ε-neighborhood graph (connecting nodes whose distances are shorter than ε), the fully connected graph (connecting all nodes with each other), and the k-nearest neighbor graph (connecting nodes u and v if u belongs to k nearest neighbors of v *or* v belongs to k nearest neighbors of u) (Luxburg 2007). The ε-neighborhood graph and k-nearest neighbor graph ($k \ll n$) are usually sparse, which have advantages in computation. The typical similarity function is the Gaussian kernel function $w_{ij} = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$ where σ is the kernel bandwidth.

Algorithm 1 shows that spectral clustering transforms the data from its original space to the eigenspace of the Laplacian matrix and uses k-means to cluster data in that space. The representation in the new space enhances the cluster properties in the data so that the clusters can be linearly separated. Therefore, traditional technique like k-means can easily cluster data in the new space.

We can use the normalized Laplacian matrix and its corresponding eigenvectors as the eigenspace. Shi and Malik (2000) computed the first k eigenvectors of the generalized eigensystem as the eigenspace. These eigenvectors are in fact the eigenvectors of the normalized Laplacian $L_n = D^{-1}L$ (Luxburg 2007). Ng et al. use k eigenvectors of the normalized Laplacian $L_n = D^{-1/2}LD^{-1/2}$ as the eigenspace. It then requires the normalization of each row in the new space to norm 1 (Ng et al. 2001).

2.2 Random projection

Principal Component Analysis is a typical method for dimensionality reduction (Jolliffe 2002). However, it has the complexity $O(d^3)$ due to the eigen decomposition of the data covariance matrix where d is the data dimension. Even if we can reduce the computation by computing the first few eigenvectors, it is still not practical for very high dimensional data.

Random projection is an alternative and less expensive way to reduce the dimensionality of very high dimensional data. According to Johnson–Lindenstrauss lemma, the pairwise Euclidean distances between data points are preserved if we randomly project the data onto a subspace spanned by $O(\log n)$ columns (Johnson and Lindenstrauss 1984). Therefore, the dimension of the projected space only depends on the number of data points, no matter how high the original dimension of the data is. Achlioptas later showed that the random projection can be applied using the following lemma (Achlioptas 2001).

Lemma 1 *Given $X \in \mathbb{R}^{n\times d}$, $\epsilon > 0$, and $k = O(\log n/\epsilon^2)$. Let $R_{d\times k}$ be a random matrix where each entry r_{ij} can be draw from the following probability distribution:*

$$r_{ij} = \begin{cases} +1 & \textit{with probability } \dfrac{1}{2s} \\ 0 & \textit{with probability } 1 - \dfrac{1}{s} \\ -1 & \textit{with probability } \dfrac{1}{2s} \end{cases}$$

where s represents the projection sparsity. With probability at least $1 - 1/n$*, the projection* $Y = XR$ *approximately preserves the pairwise Euclidean distances for all data points in X. More precisely,*

$$(1-\epsilon)\|u-v\|^2 \leq \|u'-v'\|^2 \leq (1+\epsilon)\|u-v\|^2$$

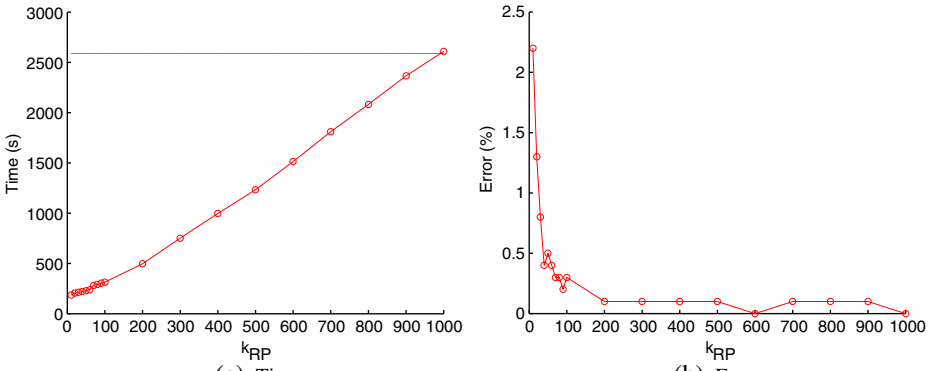
for all data points $u, v \in X$ *and* $u', v' \in Y$.

The random projection reduces the data dimension from d to $k = O(\log n)$. In practice, k can be small but still preserves the pairwise distances in the projected space.

Example to show the effectiveness of random projection, we created a synthetic dataset with $n = 10{,}000$ data points and $d = 1{,}000$ dimensions. The dataset was randomly generated from Normal distribution. Figure 2a and b show computational times for computing the average pairwise Euclidean distances of all points in the dataset with different values of k and the corresponding errors between the average pairwise distances in the original data space and the projected space. The horizontal line is the computational time in the original space. They show that random projection can significantly reduce the computational time while still maintaining accurately the distance values. The results also show that the number of columns required for random projection in practice (k_{RP}) can be quite small compared with the theoretical bound $O(\log n/\epsilon^2)$. The error curve just slightly decreases when k_{RP} reaches a certain small value.

2.3 SDD solver

Gaussian elimination is the typical method for solving linear systems. Though it can give us the exact solution to the system, its complexity of $O(n^3)$ makes it impractical



(a) Time (b) Error

Fig. 2 Random projection can significantly reduce the computational time while still maintaining accurately the distance values. The number of columns in random projection k_{RP} is quite small compared with the theoretical bound $O(\log n/\epsilon^2)$. The error curve just slightly decreases when k_{RP} reaches a certain small value

for large linear systems. Iterative methods approximate the solution gradually using only simple matrix and vector operations. Lanczos's conjugate gradient is a typical iterative method for symmetric positive definite system (Golub and Van Loan 1996). However, its running time can be large when the convergence is slow.

Vaidya is the first to suggest the idea of using graph theory for iterative methods (Vaidya 1991). Speilman and Teng used the idea to find a combinatorial preconditioner by graph sparsification and proposed the first near-linear time SDD solver (Spielman and Teng 2004). The solver applies the Chebyshev iteration with preconditioners designed using nearly-linear time algorithms for graph sparsification and graph partitioning (Spielman and Teng 2004, 2006). Recently, Koutis et al. have proposed many improvements for the construction of the preconditioners (Koutis et al. 2009, 2010, 2011).

3 Related work

Spectral clustering involves the eigen-decomposition of the Laplacian matrix which is not feasible to use in large graphs. Most of the approaches try to approximate it using sampling or low-rank approximation techniques. Fowlkes et al. (2004) used Nyström technique and Wang et al. (2009) used column sampling to solve the eigensystem in a smaller sample and extrapolated the solution for the whole dataset.

Yan et al. provided a framework for a fast approximate spectral clustering (Yan et al. 2009). A number of centers were chosen from the data by using k-means (denoted as KASP) or a random projection tree. Then these centers were clustered by the spectral clustering. The cluster membership for each data point corresponding to its center was assigned using the spectral clustering membership in the center set. However, the center selection step is time consuming for large datasets.

Chen and Cai used the idea of sparse coding to design an approximate affinity matrix $A = ZZ^T (Z \in \mathbb{R}^{n \times s}$ where s is the number of representatives, or landmarks in their word) so that the eigen decomposition of an $(n \times n)$ matrix A can be found from the eigen decomposition of a smaller $(s \times s)$ matrix $Z^T Z$ (Chen and Cai 2011). Since the smallest eigenvectors of the normalized Laplacian $L_n = D^{-1/2} L D^{-1/2}$ are the largest eigenvectors of $D^{-1/2} A D^{-1/2}$, we have the eigen solution of L_n. The s landmarks can be selected by random sampling or by k-means method. The method is denoted as LSC.

However, all these methods involve data sampling either by choosing randomly or by a k-means selection. Using k-means or other methods to select the representative centers is costly in large datasets since the number of representatives cannot be too small. Moreover, any kind of sampling will suffer from the loss of information in the original data since the representatives may not correctly capture the cluster geometry structures. Therefore, any approximation based on these representatives also suffers from this information loss. These facts will be illustrated in the experiments. Moreover, approximate methods working directly with feature data cannot be used with graph data.

Another kind of study by Mavroeidis proposed a semi-supervised framework using data labels to improve the efficiency of the power method in finding eigenvectors for spectral clustering (Mavroeidis 2010). Alternatively, Chen et al. used parallel processing to accelerate spectral clustering in a distributed environment (Chen et al.

2011). In this work, we only focus on the acceleration of spectral clustering using a single machine in an unsupervised manner.

4 Spectral clustering with SDD solvers

This section shows how to use the SDD solver and random projection to approximate spectral clustering without sampling and directly computing the eigenvectors. The approach makes use of the fact that the eigenspace is an embedding of the resistance distance.

Lemma 2 *Assuming the eigen-decomposition of the Laplacian of graph G is $L = VSV^T$ where V contains column eigenvectors and S is the diagonal matrix containing corresponding eigenvalues, $\theta_1 = VS^{-1/2} \in \mathbb{R}^{n\times n}$ is an eigenspace where the squared Euclidean distance is the resistance distance in G.*

Proof Equation (1) can be written as:

$$\begin{aligned} r_{ij} &= (e_i - e_j)^T L^+ (e_i - e_j) \\ &= (e_i - e_j)^T VS^{-1}V^T (e_i - e_j) \\ &= (e_i - e_j)^T VS^{-1/2}S^{-1/2}V^T (e_i - e_j) \\ &= [S^{-1/2}V^T (e_i - e_j)]^T [S^{-1/2}V^T (e_i - e_j)]. \end{aligned}$$

Therefore, r_{ij} is the squared Euclidean distance between i and j in the eigenspace $\theta_1 = VS^{-1/2}$. □

Fact 1 *Let m be the number of edges in an undirected and weighted graph G. If the edges in G are oriented arbitrarily, $B_{m\times n}$ given by:*

$$B(u, v) = \begin{cases} 1 & \text{if } v \text{ is } u\text{'s head} \\ -1 & \text{if } v \text{ is } u\text{'s tail} \\ 0 & \text{otherwise} \end{cases}$$

is a signed edge-vertex incidence matrix and $W_{m\times m}$ is a diagonal matrix whose entries are the edge weights. Then $L = B^T WB$.

Lemma 3 (Spielman and Srivastava 2008) *$\theta_2 = L^+ B^T W^{1/2} \in \mathbb{R}^{n\times m}$ is an embedding where the squared Euclidean distance is the resistance distance in G.*

Proof Equation (1) can be written as:

$$\begin{aligned} r_{ij} &= (e_i - e_j)^T L^+ (e_i - e_j) \\ &= (e_i - e_j)^T L^+ LL^+ (e_i - e_j) \\ &= (e_i - e_j)^T L^+ B^T WBL^+ (e_i - e_j) \\ &= [(e_i - e_j)^T L^+ B^T W^{1/2}][W^{1/2}BL^+ (e_i - e_j)] \\ &= [W^{1/2}BL^+ (e_i - e_j)]^T [W^{1/2}BL^+ (e_i - e_j)] \end{aligned}$$

Thus the r_{ij} is the squared Euclidean distance between i and j in space $\theta_2 = L^+ B^T W^{1/2}$. □

Therefore the pairwise Euclidean distances in θ_1 and θ_2 are the same. Consequently, k-means in θ_1 and θ_2 give the same clustering results. Since θ_1 is the eigenspace of the graph Laplacian, applying k-means in θ_2 is spectral clustering.

The embedding $\theta_2 = L^+ B^T W^{1/2}$ is costly to create since it takes $O(n^3)$ for the pseudo-inversion of L. We adopt the idea in Spielman and Srivastava (2008) to approximate the embedding θ_2 more efficiently.

The Euclidean distances in θ_2 (i.e. r_{ij}) are preserved using the random projection scheme described in Lemma 1. We can use a random matrix $Q_{k_{RP} \times m}$ where $Q(i, j) = \pm 1/\sqrt{k_{RP}}$ with equal probabilities.

Theorem 1 (Spielman and Srivastava 2008) *Given $\epsilon > 0$ and a matrix $Z_{O(\log n/\epsilon^2) \times n} = QW^{1/2} BL^+$, with probability at least $1 - 1/n$:*

$$(1 - \epsilon) r_{ij} \leq \| Z(e_i - e_j) \|^2 \leq (1 + \epsilon) r_{ij}$$

for all pairs $i, j \in G$.

Proof The proof comes directly from Lemmas 1 and 3. □

Therefore we are able to construct a matrix $Z = QW^{1/2} BL^+$ which $r_{ij} \approx \| Z(e_i - e_j) \|^2$ with an error ϵ. Since to compute L^+ directly is expensive, the SDD solver is used instead. First, $Y = QW^{1/2} B$ is computed. Then each of $k_{RP} = O(\log n)$ rows of Z (denoted as z_i) is computed by solving the system $z_i L = y_i$ where y_i is a row of Y. The SDD solver takes only $\widetilde{O}(m)$ to solve the system (Spielman and Srivastava 2008).

Since $\| z_i - \widetilde{z}_i \|_L \leq \epsilon \| z_i \|_L$ where $\widetilde{z}_i$ is the solution of $z_i L = y_i$ using the SDD solver (Spielman and Srivastava 2008) we have:

$$(1 - \epsilon)^2 c_{ij} \leq \| \widetilde{Z}(e_i - e_j) \|^2 \leq (1 + \epsilon)^2 c_{ij} \tag{2}$$

where $\widetilde{Z}$ is the matrix containing row vector $\widetilde{z}_i$. Equation (2) shows that the approximation has the error ϵ^2.

We propose an approximate spectral clustering method called Resistance Embedding Spectral Clustering (RESC). The key idea of the method is the creation of the eigen space via the resistance distance embedding and the effective method to approximate this embedding using random projection and the SDD solver. The method is detailed in Algorithm 2.

In Algorithm 2, $\theta = \widetilde{Z}^T \in \mathbb{R}^{n \times k_{RP} = O(\log n)}$ is the embedding space where the squared Euclidean distance is the approximate resistance distance. Applying k-means in θ is a novel way to accelerate spectral clustering without using sampling and direct eigen-decomposition.

4.1 Analysis

Here we analyze the computational complexity of the proposed method. Firstly the k_1-nearest neighbor graph is constructed in $O(n \log n)$ time using kd-tree. $Y = QW^{1/2} B$ is computed in $O(2mk_{RP} + m) = O(mk_{RP})$ time since there are only $2m$ nonzeros in B and W is a diagonal matrix with m nonzeros. Then each of k_{RP} rows

Algorithm 2 Resistance Embedding Spectral Clustering (RESC)

Input: Data matrix $X \in \mathbb{R}^{n \times d}$, number of clusters k, number of random vectors k_{RP}
Output: Cluster membership for each data point

1: Construct a k_1-nearest neighbor graph G from X ($k_1 \ll n$).
2: Compute matrices B, W, and L from G.
3: Compute $Y = QW^{1/2}B$ where Q is an $\pm 1/\sqrt{k_{RP}}$ random matrix.
4: Compute all rows $\widetilde{z}_i$ of $\widetilde{Z}_{k_{RP} \times n} = YL^+$ by k_{RP} calls to the SDD solver.
5: Cluster the points in $\widetilde{Z}^T$ using k-means.

of $\widetilde{Z}$ (denoted as $\widetilde{z}_i$) is computed by solving the system $z_i L = y_i$ in $\widetilde{O}(m)$ time where y_i is a row of Y. Since we use k_1-nearest neighbor graph where $k_1 \ll n$, $O(m) = O(n)$. Therefore, the construction of $\widetilde{Z}$ takes $\widetilde{O}(nk_{RP})$ time. k-means algorithm in $\widetilde{Z}^T$ takes $O(tkk_{RP}n)$ where k is the number of clusters and t is the number of iterations for the algorithm to be converged.

Table 1 summarizes the complexity analysis of RESC and other approximate spectral clustering techniques described in Section 3.

All methods create the embedded space where they use k-means to cluster the data. The dimension of the embedding of Nyström, KASP, and LSC is k—the number of clusters. For RESC, it is k_{RP}—the number of random projection columns.

Note that in practice, k_{RP} can be very small compared with $O(\log n/\epsilon^2)$. We will discuss it in the experimental section. We can choose $k_{RP} \ll n$. Moreover, the performance of the SDD solver is observed to be linear empirically instead of $\widetilde{O}(m)$ (Koutis et al. 2009). Therefore, the construction of $\widetilde{Z}$ takes only $O(nk_{RP})$ in practice.

On the contrary, the number of representatives s cannot be very small in order to correctly capture the geometry structure of the whole dataset. Therefore, the term $O(s^3)$ cannot be ignored.

4.2 Relationship with spectral clustering

Spectral clustering dates back to Donath and Hoffman (1973), which suggested to use eigenvectors of the adjacency matrix to partition a graph, and Fiedler (1973), which suggested to use the second eigenvector of the graph Laplacian to do the same task. Recent approaches in spectral clustering use k-means in the embedding formed by eigenvectors of the (normalized) Laplacian (Shi and Malik 2000; Ng et al. 2001).

Our approach approximates the eigen embedding for k-means through the connection with the resistance distance embedding $\theta = VS^{-1/2}$. The difference with

Table 1 Complexity comparisons of all approximate spectral clustering methods

Method	Sampling	Affinity matrix	Embedded space	k-means
Nyström	$O(1)$	$O(dsn)$	$O(s^3 + sn)$	$O(tk^2n)$
KASP	$O(tdsn)$	$O(ds^2)$	$O(s^3)$	$O(tk^2s)$
LSC	$O(1)$	$O(dsn)$	$O(s^3 + s^2n)$	$O(tk^2n)$
RESC	N/A	$O(dn \log n)$	$\widetilde{O}(k_{RP}n)$	$O(tkk_{RP}n)$

n, d, s, k_{RP}, k is the number of instances, features, representatives, random projection columns, and the number of clusters, respectively

standard spectral clustering is that the eigenspace in RESC is scaled by the inverse eigenvalues of the Laplacian.

Spectral clustering is actually the relaxed problem of the normalized cut (Luxburg 2007):

$$\min_U \operatorname{tr}(U^T L U) \text{ subject to } U^T U = I$$

The solution U is given by the first k eigenvectors of L as columns. Likewise, the resistance embedding also minimizes the normalized cut provided that the eigenvectors are scaled by the inverse corresponding eigenvalues (Qiu and Hancock 2007).

4.3 Fast k-nearest neighbor graph using density-preserving random projection

As shown in the theoretical analysis in Section 4.1 and later in experimental results in Section 5, RESC computational time is dominated by the graph creation step. This is even worse if the datasets have very high dimension which indexing techniques like kd-tree does not work efficiently.

de Vries et al. (2012) proposed a projection-indexed nearest neighbors (PINN) technique that searching for the extended nearest neighbor sets in the reduced-dimensional space by random projection in order to create an accurate approximation for k-nearest neighbor search in the original high dimensional data. They showed that the technique can preserve the neighborhood sets with high accuracy. The details are in Algorithm 3.

Algorithm 3 Projection-Indexed Nearest Neighbor (PINN)

Input: Data matrix $X \in \mathbb{R}^{n \times d}$, numbers of nearest neighbors k and $h \geq k$ in the original and projected spaces respectively, number of random vectors k_{RP}
Output: k-nearest neighbor sets for all data instances in X

1: Project X to matrix $Y \in \mathbb{R}^{n \times k_{RP}} (d > k_{RP})$ using the random projection scheme described in Lemma 1.
2: **for** each point $p \in X$ **do**
3: Denote p' be the projected point of p in Y. Find h-nearest neighbor set of p', denoted as $N_h^Y(p')$.
4: Map the points in $N_h^Y(p')$ back to the original space X, forming the set $N_h^X(p)$.
5: Find k-nearest neighbor set of p from $N_h^X(p)$, forming $N_k^X(p)$.
6: **end for**

5 Experimental results

5.1 Evaluation criteria

The experiments were carried out to determine and compare the effectiveness of the Nyström, KASP, LSC, and RESC methods. It included the clustering accuracy (percentage) and the computational time (second). For accuracy, it was measured against spectral clustering as the benchmark method since all of them are its approximations. For each dataset, the same number of clusters was used for all the clustering

methods. The accuracy was computed by counting the fraction of matching between cluster memberships of spectral clustering and the approximate method, given by:

$$Accuracy = \frac{\sum_{i=1}^{n} \delta[map(c_i) = label(i)]}{n},$$

where n is the number of data instances, $label(i)$ and c_i are the actual and the predicted cluster labels of a data instance i, respectively. δ is an indicator function and $map(c_i)$ is a permutation function that maps cluster c_i to a category label. The best matching can be found using Hungarian algorithm (Chen et al. 2011).

5.2 Methods and parameters

All the experimental results reported in the following sections were the average over ten trials. The Gaussian kernel $w_{ij} = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$ was used as the node similarity where the bandwidth σ was chosen based on the width of the neighborhood for each dataset. Methods using the nearest neighbor graph as the similarity graph chose $k_1 = 10$ as the number of nearest neighbors. For Nyström, KASP, and LSC, the eigenspaces were created from the normalized Laplacian $L_n = D^{-1/2} L D^{-1/2}$ since the normalized one is reported to be better (Luxburg et al. 2004).

All the methods were implemented using Matlab. The followings are the detailed information regarding the implementation for each method:

k-means The Matlab function 'kmeans' with 'cluster' option (i.e. cluster 10 % of the dataset to choose initial centroids) was used. The number of replications was five and the maximum number of iterations was 100.

Spectral clustering We implemented the method in Ng et al. (2001). Since it is not possible to do the eigen-decomposition of the Laplacian of a large fully connected graph, a sparse k_1-nearest neighbor graph was built and the sparse function 'eigs' was used to find the eigenspace.

Nyström We used the implementation in Chen et al. (2011) which is available online at http://alumni.cs.ucsb.edu/~wychen/sc.html.

KASP We implemented the method in Yan et al. (2009) and used k-means to select the representatives.

LSC We used the number of nearest neighbors $k_1 = 10$ for building the sparse matrix Z. In Chen and Cai (2011), the representatives can be chosen by sampling or by k-means. Since the authors claimed that the random selection had a better balance between running time and accuracy, we only used this option in the experiments.

RESC The number of random vectors $k_{RP} = 50$ was used throughout the experiments. We used the Koutis's CMG solver (Koutis et al. 2009) as an implementation of the SDD solver for creating the embedding. It is available online at http://www.cs.cmu.edu/~jkoutis/cmg.html.

5.3 An example

A synthetic dataset containing clusters featuring 'DS 2012' as in Fig. 3 was created. It has 2,000 data points in six clusters. We applied RESC, Nyström, KASP, and LSC on it. The number of representatives was 500 which was 25 % of the data. In figures of Nyström, KASP, and LSC, the red dots are the representatives selected in their corresponding methods.

It can be seen from the results the weakness of sampling-based approximate methods and the strength of RESC. Although the number of representatives was large enough (25 % of data), it did not correctly capture the geometry structures of all clusters and thus there were splits in some characters which a part of the character was considered closer to other character due to the structure of the representatives. RESC on the other hand clustered all data points correctly since it used all the data information. The exact spectral clustering also clustered the dataset correctly.

5.4 Real datasets

We tested all the four methods in several real datasets with various sizes, dimensions and number of clusters obtained from the UCI machine learning repository (Frank and Asuncion 2010). The details of all datasets are in Table 2. For all datasets, all features were normalized to have mean 0 and standard deviation 1.

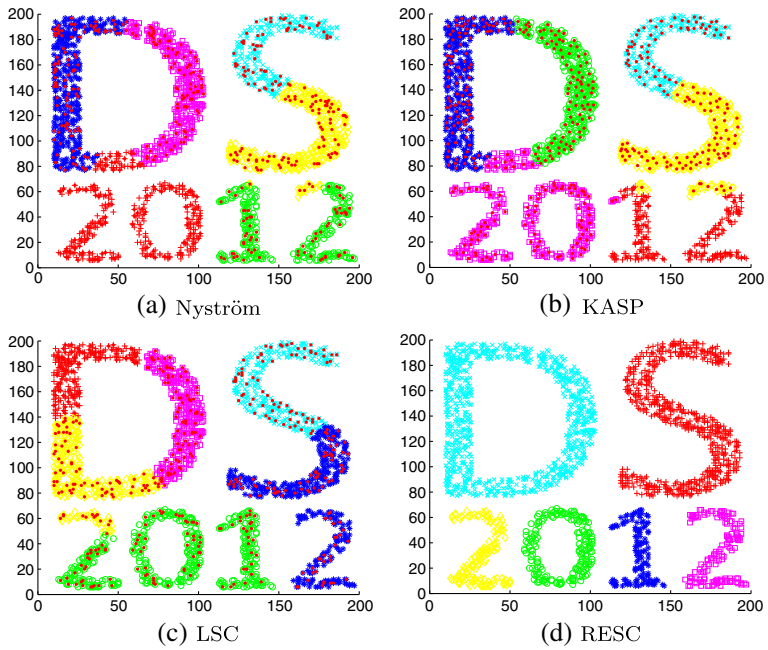


Fig. 3 Approximate spectral clustering methods using Nyström, KASP, LSC, and RESC. This shows the weakness of sampling-based approximate methods and the strength of RESC. The *red dots* are the representatives in Nyström, KASP, and LSC. Only RESC can cluster the data correctly

Table 2 UCI Datasets

Dataset	Instances	Features	Classes	Description
Segment	2,100	19	7	Image segmentation
Spambase	4,601	57	2	Spam prediction
Musk	6,598	166	2	Molecule prediction
Pen digits	10,992	16	10	Pen-based recognition of handwritten digits
Letter rec	20,000	16	26	Letter recognition
Connect4	67,557	42	3	The game of connect-4

Regarding the number of representatives, it cannot be too small to truly represent the data or too big to significantly slower the methods. For the first three small datasets (Segment, Spambase, and Musk), we used 20 % of the data as the representatives. Medium sizes Pen Digits and Letter Rec used 10 % of the data as the representatives. For the big size dataset Connect4, we only chose 5,000 data instances as the representatives. It is less than 10 % of data in Connect4. Since the computation of sampling-based methods is very expensive for large datasets if a high number of representatives is used, the percentages of the representatives used were less in larger datasets.

Tables 3 and 4 show the clustering results in accuracy (percentage) and running time (second) for all the datasets. Considering the accuracy, RESC outperformed all the sampling-based approximate methods in most of datasets although the number of representatives these methods used was high enough. Considering the computational time, RESC was the fastest method in all datasets.

From the complexity analysis in Section 4.1, we can see that the bottleneck of RESC is the total running time of graph creation and k-means steps. This is clearly shown in the results in Table 5, which presents the details in percentage of the running time of RESC for each dataset. The running time of the embedding step was dominated by the other two steps. The advantage is that there have been many studies in fast k-means and graph creation, or techniques to parallel them which we can make use of Chen et al. (2011). On the contrary the main cost of KASP, Nyström, and LSC involves the creating of the eigenspace which is already accelerated.

5.5 Parameter sensitivity

As we have already mentioned, k_{RP} is small in practice and there is not much differences between different datasets. Venkatasubramanian and Wang (2011) suggested that $k_{RP} = 2 \ln n / 0.25^2$ which is just about 500 for a dataset of ten millions

Table 3 Clustering accuracy (percentage)

RESC outperformed other methods in most of the datasets

Dataset	KASP (%)	Nyström (%)	LSC (%)	RESC (%)
Segment	74.5	58.3	73.2	**78.9**
Spambase	60.8	82.7	97.6	**100**
Musk	81.3	50.6	63.2	**97.2**
Pen Digits	**83.4**	74.8	80.1	77.5
Letter Rec	52.8	39.2	**58.5**	40.1
Connect4	86.8	35.3	83.0	**97.4**

Table 4 Computational time (second)

Dataset	KASP	Nyström	LSC	RESC
Segment	2.26	6.25	8.87	**2.06**
Spambase	25.33	28.26	46.68	**16.32**
Musk	178.24	110.87	154.09	**63.18**
Pen digits	65.33	104.01	119.04	**12.46**
Letter rec	236.43	529.86	395.47	**59.45**
Connect4	3400.38	10997.14	3690.86	**1839.59**

RESC was the fastest among all the approximate methods

points. We conducted an experiment with different k_{RP} in each dataset. The results in Fig. 4 show that the parameter k_{RP} is quite small compared with the theoretical bound $O(\log n/\epsilon^2)$ (other datasets also had similar tendency). It shows that our $k_{RP} = 50$ was suitable for the datasets in the experiments. Moreover, the experiment in last section shows that the computational time of the embedding step of RESC is dominated by the other steps. Therefore, k_{RP} can be quite small and does not considerably affect the running time of RESC. This is another advantage of RESC since it is not sensitive to the parameters in terms of both accuracy and performance. For the sampling-based methods, the selection of the number of representatives to balance between accuracy and speed is not trivial.

5.6 Graph datasets

One more advantage of RESC over other approximate methods is that it can work directly on the similarity graph while the others cannot if they work directly on the original feature data. An experiment to show the scalability of the proposed method in large graphs was conducted in DBLP co-authorship network obtained from http://dblp.uni-trier.de/xml/ and some real network graphs obtained from the Stanford Large Network Dataset Collection which is available at http://snap.stanford.edu/data/. *CA-AstroPh* is a collaboration network of Arxiv Astro Physics; *Email-Enron* is an email communication network from Enron company; and *RoadNet-TX* is a road network of Texas in the US.

All the graphs were undirected. The largest connected component was extracted if a graph was not connected. We arbitrarily chose $k = 50$ as the number of clusters for all the datasets. The results using RESC are shown in Table 6.

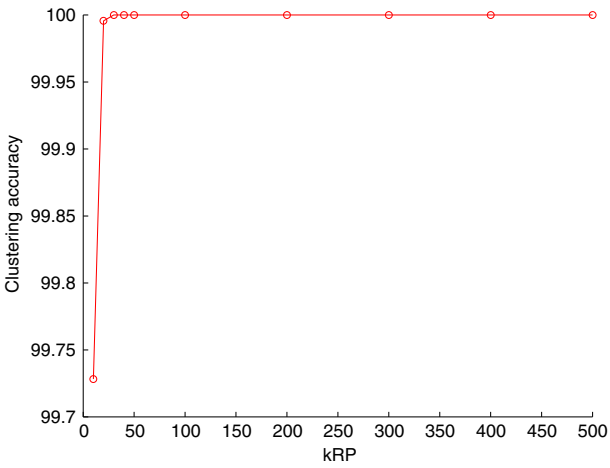
In case of graph data, the running time of k-means was dominant the whole method. RESC took only less than 10 min to create an approximate embedding for a network graph of more than 1.3 million nodes.

Since all the above graphs are too big to do a qualitative analysis, a subset of main data mining conferences in the DBLP graph was analyzed. We selected only authors and publications appearing in KDD, PKDD, PAKDD, ICDM, and SDM.

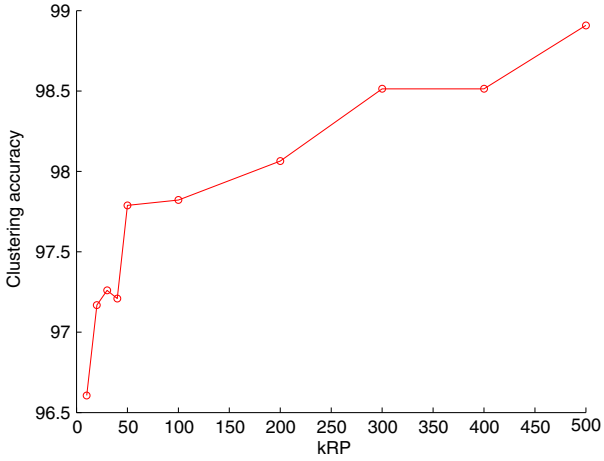
Table 5 Time distribution for RESC

Datasets	Graph (%)	Embedding (%)	k-means (%)
Segment	54.0	31.1	14.9
Spambase	90.1	9.1	0.8
Musk	92.6	6.9	0.5
Pen digits	51.1	33.0	15.8
Letter rec	36.5	17.0	46.5
Connect4	97.1	2.7	0.2

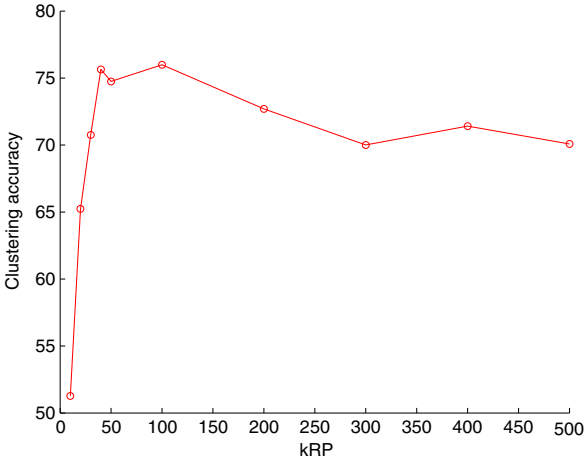
The bottleneck of the algorithm is the total running time of graph creation and k-means steps



(a) Spambase



(b) Musk



(c) Pen Digits

Fig. 4 The number of columns (k_{RP}) required in practice for random projection is much smaller than the theoretical bound $O(\log n/\epsilon^2)$

Table 6 The clustering time (second) for some network graphs

Dataset	Nodes	Edges	Embedding	k-means	Total time (s)
CA-AstroPh	17,903	197,001	24.36	50.62	74.98
Email-Enron	33,696	180,811	27.33	167.08	194.41
DBLP	612,949	2,345,178	764.04	4572.25	5336.31
RoadNet-TX	1,351,137	1,879,201	576.62	4691.53	5268.15

RESC took less than 10 min to create an approximate embedding for a network graph of more than 1.3 million nodes

Each author also need to have at least ten publications and his/her co-authors also need to have such minimum publications. This selected only authors who published highly in major data mining conferences and collaborated with the similar kind of co-authors. Then the biggest connected component of the graph was extracted. The final graph has 397 nodes and 1,695 edges. The average node degree is 4.27 and the average node weight is 10.48.

RESC was applied to the subgraph with $k = 50$ clusters. Since researchers have collaborated and moved from research groups to research groups overtime, some clusters are probably a merge of groups caused by the collaborations and movings of prominent researchers. However, the method can effectively capture clusters representing many well known data mining research groups in CMU, IBM Research Centers (Watson and Almaden), University of California Riverside, LMU Munich, University of Pisa, University of Technology Sydney, Melbourne University, etc.

5.7 Fast k-nearest neighbor graph using random projection

As noted earlier, graph creation is a bottleneck of the running time of RESC, especially for high dimensional data. In this section, experiments in high dimensional datasets were carried out using the method described in Section 4.3 to accelerate the graph creation step. Some high dimensional datasets used in previous experiments and more datasets described in Table 7 were used to evaluate the effectiveness of the method. Covertype was from UCI repository (Frank and Asuncion 2010) and the others were obtained from http://www.cad.zju.edu.cn/home/dengcai/Data/data.html. Reuters dataset originally contains 21,578 documents in 135 categories. We removed documents with multiple category and category which has less than 100 documents. It left us with 7,195 documents in nine categories.

Tables 8 and 9 compare the clustering accuracy and computational time between the original RESC and RESC with approximate k-nearest neighbor graph using random projection (denoted as RESC + RP). We used $k = 10$, $h = 3k$, and $k_{RP} = 20$

Table 7 High dimensional datasets

Dataset	Instances	Features	Classes	Description
COIL20	1,440	1,024	20	Columbia Object Image Library
Reuters	7,195	18,933	9	Document clustering
MNIST	70,000	784	10	Hand written digits
Covertype	581,012	54	7	Forest cover type prediction

Table 8 Clustering accuracy (percentage)

Dataset	RESC (%)	RESC + RP (%)
Spambase	99.9	98.6
Musk	98.6	95.3
COIL20	72.1	70.6
Reuters	12.8	12.7
Connect4	94.7	89.0
MNIST	15.1	11.9
Covertype	77.4	67.2

RESC + RP well maintained the clustering accuracy

for RESC + RP as suggested in de Vries et al. (2012). Note that k_{RP} here is different from the parameter of random projection used in the approximation of the eigen embedding of RESC.

The results show that random projection can significantly reduce the time for graph creation step which is the bottleneck of RESC while still maintaining the clustering accuracy. When the dimension was very high (in Musk, COIL20, Reuters, and MNIST), RESC + RP was much faster than RESC. The only exception is in Spambase dataset where RESC + RP was slightly slower than RESC. The reason was the dataset size and dimension are not big enough so that the cost of random projection step is higher than the benefit of computation in the slightly reduced-dimensional space.

5.8 Discussion

Von Luxburg, Radl, and Hein in their paper (von Luxburg et al. 2010) showed that the resistance distance between two nodes on a random geometric graph converges to an expression that only depends on the degrees of these two nodes and does not take into account the structure of the graph. Specifically, $r_{ij} \approx \frac{1}{d_i} + \frac{1}{d_j}$. Therefore, they claimed that the resistance distance is often misleading as a distance function on large graphs. However, their results do not reject our work because of the following reasons.

– Their proof was based on random geometric graphs which may not be the case in all datasets. Moreover, their claim can only hold under some assumptions.

Table 9 Clustering time (s)

Dataset	Graph creation		Total time	
	RESC	RESC + RP	RESC	RESC + RP
Spambase	10.64	12.26	12.17	13.88
Musk	51.78	18.03	53.50	19.72
COIL20	14.49	4.15	15.22	4.96
Reuters	6,206.43	255.62	6,214.65	264.66
connect4	1,807.59	712.11	1,833.48	739.74
MNIST	23,419.14	1,432.67	23,553	1,572.35
Covertype	5,255.91	2,088.09	5,505.52	2,345.20

RESC + RP was much faster than RESC

Table 10 Comparison between RESC and the clustering using the approximation in von Luxburg et al. (2010)

The results show that the resistance distance is not meaningless to use as a robust metric for our approximate spectral clustering

Datasets	RESC (%)	Clutering in von Luxburg et al. (2010) (%)
Segment	79.1	31.9
Spambase	100	96.6
Musk	97.7	86.0
Pen digits	81.9	16.7
Letter rec	41.2	17.8
Connect4	93.9	80.0

For example, their approximation only holds if the minimal degree in the graph increases with the number of nodes in the graph.

– Their experiments showed that the approximation becomes worse when the data have cluster structure. However, a condition for any unsupervised distance-based technique can work well is the data should have a cluster structure so that the separation based on distance is meaningful. We believe that many real datasets should have cluster structures in a certain degree.
– Our experiments show that RESC has good clustering results in several synthetic and real datasets. It shows that the proposed method can still be potential for using as a fast and accurate approximate spectral clustering.

To be more convincing, we conducted an experiment to compare between the clustering using the approximation of resistance distance in von Luxburg et al. (2010) and our RESC. Standard spectral clustering was used as the benchmark and the clustering accuracies of the two methods in datasets used in the previous section were reported. The results in Table 10 show RESC had better clustering results than these of the clustering using the approximation in von Luxburg et al. (2010). This shows the resistance distance is not meaningless to use as a robust metric for our approximate spectral clustering.

One note here is the approximation $r_{ij} \approx \frac{1}{d_i} + \frac{1}{d_j}$ is not completely meaningless in some situations. As in the k nearest neighbor graph, nodes in a cluster with a similar density distribution is likely to have a similar degree, the clustering using this approximation may behave well in case all the clusters have different density distributions.

6 Conclusion

The paper shows a fast and accurate approximation for spectral clustering. The strength of the method is that it does not involve any sampling technique which may not correctly represent the whole dataset. It does not need to directly compute any eigenvector as well. Instead it reformulates the spectral clustering problem using the resistance distance embedding and uses the random projection and the SDD solver to approximate the embedding. The experimental results in several synthetic and real datasets and graphs with various sizes show the effectiveness of the proposed approaches in terms of performance and accuracy. It is faster than the state-of-the-art approximate spectral clustering techniques while maintaining better clustering accuracy. The proposed method can also be applied directly to graph data.

Though the analysis and experimental results show that RESC and spectral clustering have quite similar clustering ability, a deeper theoretical analysis need to be done to examine the strength and weakness of each method.

References

Achlioptas, D. (2001). Database-friendly random projections. In *Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART symposium on Principles of Database Systems, PODS '01* (pp. 274–281). New York: ACM.

Chen, X., & Cai, D. (2011). Large scale spectral clustering with landmark-based representation. In *Twenty-Fifth AAAI Conference on Artificial Intelligence* (pp. 313–318).

Chen, W.Y., Song, Y., Bai, H., Lin, C.J., Chang, E.Y. (2011). Parallel spectral clustering in distributed systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 33*(3), 568–586.

de Vries, T., Chawla, S., Houle, M.E. (2012). Density-preserving projections for large-scale local anomaly detection. *Knowledge and Information Systems, 32*(1), 25–52.

Donath, W.E., & Hoffman, A.J. (1973). Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development, 17*, 420–425.

Doyle, P.G., & Snell, J.L. (1984). *Random walks and electric networks*. Washington, DC: Mathematical Association of America.

Fiedler, M. (1973). Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal, 23*, 298–305.

Fouss, F., & Renders, J.M. (2007). Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transaction on Knowledge and Data Engineering, 19*(3), 355–369.

Fowlkes, C., Belongie, S., Chung, F., Malik, J. (2004). Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 26*, 214–225.

Frank, A., & Asuncion, A. (2010). UCI machine learning repository. URL: http://archive.ics.uci.edu/ml. Accessed 31 Jan 2013

Golub, G.H., & Van Loan, C.F. (1996). *Matrix computations* (3rd edn.). Baltimore: Johns Hopkins University Press.

Johnson, W., & Lindenstrauss, J. (1984). Extensions of Lipschitz mappings into a Hilbert space. In *Conference in modern analysis and probability* (New Haven, Conn., 1982), *Contemporary Mathematics* (Vol. 26, pp. 189–206). American Mathematical Society.

Jolliffe, I.T. (2002). *Principal component analysis* (2nd edn.). Springer.

Koutis, I., Miller, G.L., Tolliver, D. (2009). Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing. In *Proceedings of the 5th international symposium on advances in visual computing: Part I, ISVC '09* (pp. 1067–1078). Berlin, Heidelberg: Springer.

Koutis, I., Miller, G., Peng, R. (2010). Approaching optimality for solving sdd linear systems. In *2010 51st annual IEEE symposium on Foundations of Computer Science (FOCS)* (pp. 235–244).

Koutis, I., Miller, G.L., Peng, R. (2011). A nearly-m log n time solver for sdd linear systems. In *Proceedings of the 2011 IEEE 52nd annual symposium on Foundations of Computer Science, FOCS '11* (pp. 590–598). Washington, DC: IEEE Computer Society.

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

Luxburg, U.V., Bousquet, O., Belkin, M. (2004). On the convergence of spectral clustering on random samples: The normalized case. In *Proceedings of the 17th annual Conference on Learning Theory (COLT)* (pp. 457–471). Springer.

von Luxburg, U., Radl, A., Hein, M. (2010). Getting lost in space: Large sample analysis of the resistance distance. In *NIPS* (pp. 2622–2630).

Mavroeidis, D. (2010). Accelerating spectral clustering with partial supervision. *Data Mining and Knowledge Discovery, 21*, 241–258.

Ng, A.Y., Jordan, M.I., Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems* (pp. 849–856). MIT Press.

Qiu, H., & Hancock, E. (2007). Clustering and embedding using commute times. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 29*(11), 1873–1890.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*, 888–905.

Spielman, D.A., & Srivastava, N. (2008). Graph sparsification by effective resistances. In *Proceedings of the 40th annual ACM Symposium on Theory of Computing, STOC '08* (pp. 563–568). New York: ACM.

Spielman, D.A., & Teng, S.H. (2004). Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th annual ACM Symposium on Theory of Computing, STOC '04* (pp. 81–90). New York: ACM.

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

Vaidya, P. (1991). Solving linear equations with symmetric diagonally dominant matrices by constructing good preconditioners. A Talk Based on this Manuscript was Presented at the IMA Workshop on Graph Theory and Sparse Matrix Computation, October 1991, Minneapolis.

Venkatasubramanian, S., & Wang, Q. (2011). The Johnson–Lindenstrauss transform: An empirical study. In M. Müller-Hannemann, R.F.F. Werneck (Eds.), *ALENEX* (pp. 164–173). SIAM.

Wang, L., Leckie, C., Ramamohanarao, K., Bezdek, J. (2009). Approximate spectral clustering. In *Proceedings of the 13th Pacific-Asia conference on advances in knowledge discovery and data mining, PAKDD '09* (pp. 134–146). Berlin Heidelberg: Springer.

Yan, D., Huang, L., Jordan, M.I. (2009). Fast approximate spectral clustering. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge Discovery and Data mining, KDD '09* (pp. 907–916). New York: ACM.