SCAR — Spectral Clustering Accelerated and Robustified

Ellen Hohma*
Technical University of Munich
Munich, Germany
ellen.hohma@tum.de

Christian M.M. Frey*
Christian-Albrecht University of Kiel
Kiel, Germany
cfr@informatik.uni-kiel.de

Anna Beer*
Aarhus University
Aarhus, Denmark
beer@cs.au.dk

Thomas Seidl
LMU Munich
Munich, Germany
seidl@dbs.ifi.lmu.de

ABSTRACT

Spectral clustering is one of the most advantageous clustering approaches. However, standard Spectral Clustering is sensitive to noisy input data and has a high runtime complexity. Tackling one of these problems often exacerbates the other. As real-world datasets are often large *and* compromised by noise, we need to improve both robustness and runtime at once. Thus, we propose **S**pectral **C**lustering - **A**ccelerated and **R**obust (SCAR), an accelerated, robustified spectral clustering method. In an iterative approach, we achieve robustness by separating the data into two latent components: cleansed and noisy data. We accelerate the eigendecomposition – the most time-consuming step – based on the Nyström method. We compare SCAR to related recent state-of-the-art algorithms in extensive experiments. SCAR surpasses its competitors in terms of speed and clustering quality on highly noisy data.

PVLDB Reference Format:
Ellen Hohma, Christian M.M. Frey, Anna Beer, and Thomas Seidl. SCAR — Spectral Clustering Accelerated and Robustified. PVLDB, 15(11): 3031 - 3044, 2022.
doi:10.14778/3551793.3551850

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/SpectralClusteringAcceleratedRobust/SCAR.

1 INTRODUCTION

Clustering is a fundamental data mining task needed in virtually all areas working with data and also serves as an unsupervised preprocessing step for a plethora of subsequent tasks. One of the most favorable clustering methods is spectral clustering: it is applicable to non-numeric datasets, can find clusters of complex shapes and different densities, and optimizes a mathematically well-defined problem [52]. However, real-world datasets are challenging for several reasons: with newly developed data gathering methods (e.g., in medicine, chemistry, or biology), in recent years datasets grew in dimensionality as well as in size. The runtime complexity of spectral clustering methods is only linear in the number of dimensions, as it works on an affinity graph of the data, making it superior to more traditional clustering methods when working on high-dimensional data. However, the runtime complexity (for the naive implementation) of $O(n^3)$ w.r.t. the number of data points is comparably large. Furthermore, real-world data often contains noise that is neither handled well by standard spectral clustering methods nor by other clustering methods. Clustering noisy data is in fact a very challenging task, as Fig. 1 illustrates. It shows a very noisy version of the well-known synthetic moons dataset as clustered by diverse algorithms. It may not come as a surprise that standard Spectral Clustering fails to detect the moons correctly. But also state-of-the-art algorithms that are designed specifically to be robust against noise can only handle noise up to a certain degree and were not able to detect the two clusters correctly. Our competitors in Fig. 1 (as well as in our experiments in Sec. 5) are recent clustering algorithms published at high-quality conferences: RSC [7], DCF [50], and SpectACl [21]. The authors of all methods performed extensive experiments showing their superiority against a variety of other clustering methods regarding noise robustness. RSC and DCF also successfully tackle the efficiency problems of clustering. Nevertheless, with our newly developed method SCAR (**S**pectral **C**lustering - **A**ccelerated and **R**obust), we found a way to even further improve both, clustering quality on highly noisy data *and* efficiency on high-dimensional data.

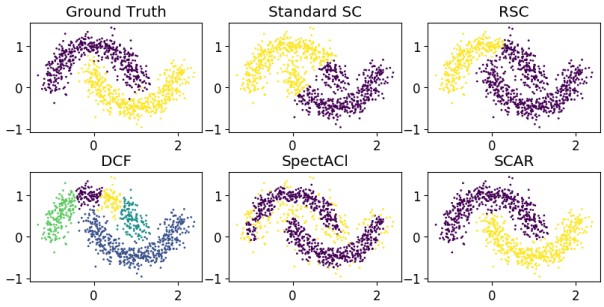


Figure 1: Our method SCAR vs. state-of-the art related clustering algorithms on the moons dataset with *noise* = 0.15.

SCAR uses weighted kNN graphs to capture highly complex structures in the data implying clusters of non-convex shapes. For a good segmentation of the graph, normalized cuts have proven to be

*Authors contributed equally to this work.

Proceedings of the VLDB Endowment, Vol. 15, No. 11 ISSN 2150-8097.
doi:10.14778/3551793.3551850

desirable [10, 12], suggesting a spectral approach. Based on the concept of RSC [7], we divide the data into a subset containing noise and a subset containing the relevant information for clustering. However, RSC involves the frequent calculation of eigendecompositions in an iterative approach, which we accelerate with the Nyström method. With an elaborated combination of synergistic methods and changes we manage to achieve highly competitive results regarding the clustering quality and robustness. In extensive and reproducible experiments we examine and compare our clustering results w.r.t. quality and runtime. SCAR shows the desired behavior for highly noisy datasets, where it outperformed recent state-of-the-art algorithms in quality, noise robustness, and runtime. We evaluated diverse types of noise and used well-known benchmark datasets. Our main contributions are as follows:

- We introduce SCAR, our novel spectral clustering method tackling both, robustness *and* speed.
- We incorporate the Nyström method to accelerate the eigendecomposition in robust spectral clustering.
- We further enhance quality and stability of clusterings
- We evaluate our method thoroughly, fairly, and reproduciblyand compare our method to recent state-of-the-art methods on the established real-world benchmark datasets.

Outline. In Sec. 2 we give an overview on related methods. In Sec. 3 we explain the basics for our new method. In Sec. 4 we introduce our new fast and robust spectral clustering method, called SCAR. In Sec. 5 we evaluate SCAR thoroughly, objectively, and reproducibly. Sec. 6 concludes this paper.

2 RELATED WORK

Spectral clustering refers to a set of clustering algorithms that partition a given dataset based on the spectrum of the datapoints' affinity matrix. They essentially follow three steps [52]: (1) construct a similarity graph $\mathcal{G}$, (2) compute the Laplacian of $\mathcal{G}$ and its eigendecomposition, and (3) cluster its eigenvectors with a standard clustering method, e.g., k-Means [36, 37]. Spectral clustering surpasses traditional clustering techniques in several aspects: e.g., they find arbitrarily shaped clusters, are applicable on categorical data, solve a clearly defined mathematical goal [52], and can handle varying densities. However, spectral clustering is noise-sensitive [7, 21] and has a relatively high runtime. In the following, we provide an overview of related works in the research field.

2.1 Improving Runtime

Most recent advances improving any of the steps of spectral clustering can be found in [51]. In the following, we focus on approaches accelerating the most time-intensive step of spectral clustering, the eigendecomposition. The acceleration is usually achieved with one of two strategies: iteration or sampling.

Iterative approaches. The probably most common method to accelerate the computation of eigenvectors and eigenvalues of a matrix is the *power iteration*. By iteratively multiplying the matrix with a randomly initialized vector (or an estimation of the dominant eigenvector), the eigenvector belonging to the largest eigenvalue is approximated. Generally, the frequent matrix multiplications are expensive, and only the dominant eigenvector can be approximated with the original power method – for spectral clustering, the eigenvectors belonging to the *smallest* eigenvalues are of interest. Note, that the behavior of convergence of iterative approaches usually depends on the distribution and gaps between the eigenvalues [51]. There is a wealth of extensions based on the power iteration aimed at alleviating its downsides for spectral clustering. Using Krylov subspaces allows approximating several eigenvectors at once: E.g., the *Arnoldi iteration* [1] orthogonalizes the vectors spanning the Krylov subspace by applying the Gram-Schmidt process. For Hermitian matrices like symmetric Laplacians, which are used in the process of spectral clustering, the *Lanczos* method has been proposed [29]. The Lanczos method approximates the largest k eigenvectors in $O(|\mathcal{E}|k + |\mathcal{V}|k^2)$ for a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ [51]. It is used for spectral clustering in [46]. While the Lanczos method performs well even for sparse matrices, it is often prone to numerical instability [9]. The *Implicitly Restarted Lanczos Method* (IRLM) as used, e.g., in ARPACK [31], can reduce numerical instability. Further adaptions involve among others using the inverse matrix to get the smallest eigenvalues and respective eigenvectors [15]. The Krylov-Schur algorithm [47] alleviates additional problems emerging with very large Hermitian or non-Hermitian matrices. As the convergence of symmetric cases depends on the gap ratio of the eigenvalues [41], both ends of the spectrum are approximated, e.g., with *IRLM-BE*.

Sampling-based approaches. Sampling based approaches work on (1) a subset of the edges in the similarity graph or on (2) a subset of the nodes: (1) implies a sparser Laplacian than the original one, while (2) implies a Laplacian of lower dimensionality.

Approach (1) accelerates the eigendecomposition by leveraging matrix operations that are optimized towards sparse input matrices. Working on matrices defined by kNN graphs, choosing a small k leads to sparse matrices that still hold relevant information on the structure of the data. For general graphs, *spectral sparsification* can be applied. It approximates the graph Laplacian with a matrix of same size containing fewer sampled entries. The sampling process ensures that certain pre-defined properties are respected (cf. [51]).

Approach (2) includes *graph coarsening* methods (e.g., [20, 27, 51]) that reduce the original similarity graph to a coarser graph, leading to an adjacency matrix of significantly lower dimensionality. Computing the eigendecomposition on the lower-dimensional matrix and refining it afterwards leads to a significant acceleration.

In our approach introduced in Sec. 4, we use the sampling-based *Nyström method*, which is an effective method to significantly speed-up spectral clustering while maintaining good overall eigenvector accuracy (e.g., [16, 32, 53, 54]). The Nyström method has been analyzed, replicated and improved throughout multiple studies: [6], [13], [45], and [56] focus on the improvements of particular downsides, such as the partial loss of information by sampling landmark points. Furthermore, they provide theoretical evaluations and frameworks on how the quality of the resulting spectral embedding is affected by applying the Nyström approximation. In [43], the impact of the number of landmarks selected as subsample as well as the influence on the overall clustering accuracy is investigated. Thorough studies in [17], [38], and [28] show the impact of sampling techniques picked for identifying the base subset for the Nyström extension. A theoretical analysis of the algorithm's performance and derivations of error bounds are formulated in [11]. We explain the Nyström method in detail in Sec. 3.2.

2.2 Improving Noise Robustness

As spectral clustering has no inherent noise-handling, its quality can suffer from diverse types of noise that often occur in real-world data. In the following, we distinguish between four different notions of noise that are often mixed up in the literature or not clarified: (1) additional noise points, (2) jitter, (3) noisy features, and (4) noisy edges. Even though they are closely interrelated, they can imply different challenges for (spectral) clustering.

Additional noise points. The probably most common notion of noise is that there are additional noise points in the dataset. They are typically uniformly distributed (and iid) and do not belong to any cluster. E.g., NRSC [35] tackles such noise for spectral clustering by assigning all noise points to an extra cluster. However, they work on the fully connected graph and assume that the majority of edges connected to a noise point has a low weight. AHK [23] also tackles this kind of noise and simultaneously robustify spectral clustering regarding the parameter choice by using an aggregated heat kernel. CAHSM [34] use a hypergraph to compensate for outliers and noise.

Jitter. Adding noise to a dataset can also imply adding a small deviation to each point. E.g., noise adjustment for the moons datasets regulates the deviation from the "perfectly-shaped" moons. A similar effect can be achieved by data quantization. In [24], error bounds for spectral clustering on data with jitter, resp., *perturbed data* are evaluated. Robustness against this type of noise for spectral methods is evaluated, e.g., in SpectACL [21], and RSC [7].

Noisy features. Especially in high-dimensional data, we may encounter noisy features. These refer, for example, to uniformly distributed dimensions of the data that are irrelevant for clustering for at least some points. FWKE-SC [26], SSCG [18], [57], and [3] combine feature weighting with spectral clustering to tackle this problem (similarly to subspace clustering). As they mainly focus on the construction of the similarity matrix, they can be combined with our approach in future work.

Noisy edges. Noise in graphs can also occur as additional edges in the affinity graph of the data. RSC [7] (cf. Sections 2.3 and 3.3) focuses on removing edges that connect different clusters, which are also called *corrupted edges*. RSEC [49] regards noisy edges in the context of spectral ensemble clustering. In [4], noise is regarded as "an additive perturbation to the similarity matrix", including noisy edges as well as corrupted weights of existing edges.

In this paper, we focus on robustness w.r.t. noisy edges and jitter. For the other types of noise, we suggest to filter additional noise points in a preprocessing step. For noisy features, our approach can easily be combined with feature weighting approaches that adapt the initial affinity matrix, as SCAR builds on top of the affinity matrix. For weighting the importance of features, one can follow approaches like FWKE-SC [26], using the concept of knowledge entropy, or apply importance scores for attributes that adapt to every point individually, like KISS [5].

2.3 Comparative Methods

In our experiments in Sec. 5 we compare our newly developed method SCAR with standard Spectral Clustering (SC) [40] as well as high-quality state-of-the-art spectral methods that aim at robustness and efficiency: Robust Spectral Clustering (RSC) [7] and SpectACL [21]. Furthermore, we include the very recently introduced method DCF [50] into our analyses. DCF is not a spectral approach, but also aims at robustness and efficiency.

RSC jointly performs the standard Spectral Clustering and the decomposition of the adjacency matrix A. The latter is assumed to be an additive decomposition of two latent factors, a graph containing corrupted edges and a graph representing the noise-free data. As RSC outperforms basic clustering principles like k-Means and density-based clustering methods on noisy datasets [7], it serves as a baseline in our evaluation in Sec. 5.

SpectACL combines approaches from spectral clustering and DBSCAN to solve their major drawbacks regarding noise sensitivity for minimum cut clustering and varying densities for density-based clustering [21]. The core idea is to determine clusters with large average densities while optimizing the density parameters using the spectrum of the weighted adjacency matrix.

DCF aims at improving peak-finding techniques for density-based clustering, which determine groups in a dataset based on their high density as well as distances to clusters of higher density [50]. The approach applies the peak-finding criterion to determine cluster cores instead of point modes, which enables the detection of clusters with varying densities.

3 PRELIMINARIES

In the following we give some preliminary basics for our method SCAR. In Sec. 3.1 we clarify the notation used throughout our work. In Sec. 3.2 we explain the Nyström method that we use to accelerate the eigendecomposition in detail. In Sec. 3.3 we elaborate on the robustificaton method we incorporate in our method SCAR.

3.1 Notation

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ be an undirected, weighted graph where $\mathcal{V}$ denotes a set of nodes, $\mathcal{E}$ denotes a set of edges connecting nodes, and w denotes a weight function on the edges $w : \mathcal{E} \to \mathbb{R}^{>0}$. Its adjacency matrix $A \in \mathbb{R}^{n \times n}$ is defined by its entries a_{ij} with $a_{ij} = w(v_i, v_j)$ if $(v_i, v_j) \in \mathcal{E}$, else $a_{ij} = 0$. Let $D := \mathrm{diag}(\deg(v_1), \ldots, \deg(v_n)) \in \mathbb{R}^{n \times n}$ be the degree matrix of $\mathcal{G}$ where $\deg(v_i) := |\{v_j \in V \mid (v_i, v_j) \in \mathcal{E}\}|$ is the degree of node v_i. We define the Laplacian L of $\mathcal{G}$ as $L := D - A$. The Laplacian L is symmetric and positive-semidefinite in $\mathbb{R}^{n \times n}$. Hence, the n eigenvalues $\Lambda = [\lambda_1, \ldots, \lambda_n]$ of L are real and positive. The associated eigenvectors are denoted by $H = [h_1, \ldots, h_n]$, resp., the approximated eigenvectors by $\hat{H}$. Furthermore, we denote by $\mathcal{X} = \{x_i\}_{i=1}^{n}$ the set of n input data samples, where $x_i \in \mathbb{R}^d$ is a d-dimensional feature vector.

3.2 Nyström Method for Eigenvector Approximation

The Nyström method has shown great promise in existing literature to speed-up the eigenvector calculation (e.g., [16, 32, 53, 54]). To accelerate the eigenvector computation, we use only a subsample of the whole dataset. A matrix $M \in \mathbb{R}^{n \times n}$ can be partitioned into:

$$M = \begin{bmatrix} M_1 & {M_2}^T \\ M_2 & M_3 \end{bmatrix}, \tag{1}$$

where $M_1 \in \mathbb{R}^{m \times m}$ represents the affinities between m sampled points in the subset, $M_2 \in \mathbb{R}^{(n-m) \times m}$ contains all weights from

the $n-m$ remaining points to the m subsampled points and $M_3 \in \mathbb{R}^{(n-m)\times(n-m)}$ captures the remaining affinities between all points not chosen for the subset. After choosing landmark points for the approximation, the eigenvectors H_1 of M_1 can be calculated. We introduce diverse eigendecomposition approaches that can be used in Sec. 2.1 and compare them empirically in Sec. 5.6.2.

Using the Nyström extension [16], we can extrapolate the eigenvectors for all remaining points. Let H and Λ be the eigenpairs of M, it follows:

$$M = H\Lambda H^T = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix} \Lambda \begin{bmatrix} H_1 \\ H_2 \end{bmatrix}^T = \begin{bmatrix} H_1\Lambda H_1^T & H_1\Lambda H_2^T \\ H_2\Lambda H_1^T & H_2\Lambda H_2^T \end{bmatrix} \quad (2)$$

Thus, if H_1 denotes the eigenvectors for the subsampled points M_1, we can deduce H_2, the eigenvectors for all remaining points, with $H_2 = M_2 H_1 \Lambda^{-1}$. Sorting the extrapolated eigenvectors for the remaining points back into the calculated eigenvectors for points chosen as subsample yields the approximated eigenvectors $\hat{H} \in \mathbb{R}^{n\times m}$ for M:

$$\hat{H} = \begin{bmatrix} H_1 \\ M_2 H_1 \Lambda^{-1} \end{bmatrix} \quad (3)$$

In the last step, we orthogonalize the approximated eigenvectors $\hat{H}$. By using only a subsample of the data, the time complexity can be reduced from $O(n^3)$ to $O(nm^2 + m^3)$, where usually $m \ll n$ [33].

3.3 Robustifying Spectral Clustering

In order to robustify spectral clustering, we follow RSC [7]. The main idea is to separate the input graph with adjacency matrix A into two latent subcomponents described by A^g and A^c:

$$A = A^g + A^c \quad (4)$$

A^c reflects the **c**orrupted edges in the graph and A^g contains only the noise-free, "**g**ood" edges. The partitioning into two segments can be determined and improved by independently optimizing the spectral embedding for each subgraph. In practice, it is sufficient to resolve only one component, since its counterpart can easily be deduced from the original representation (see Equation 4). In [52], it has been shown that spectral clustering can be transformed into a trace minimization problem for A. Following this idea, in [7], the authors proved that the solution to A^c can be attained by solving a maximization problem for $Tr(H^T L(A^c)H)$, where $L(A^c)$ denotes the Laplacian of matrix A^c. The corresponding objective function for the unnormalized Laplacian (cf. [7]) is defined as:

$$f([a^c_e]_{e\in\mathcal{E}}) := \sum_{(v_i,v_j)\in\mathcal{E}} a^c_{i,j} \cdot \|h_i - h_j\|_2^2 \quad (5)$$

Further constraints are given by θ and m. The parameter θ denotes the maximum number of global corruptions that are deleted: $|\{(v_i,v_j)|a^c_{ij} \neq 0\}| \leq 2\cdot\theta$. The parameter m implies the minimum number of nodes that each node in A^g is connected to: $|\{v_j|a^g_{ij} \neq 0\}| \geq m \cdot deg(v_i)$ for each node v_i.

To solve the maximization problem in order to find edges which should be assigned to A^c, we use a greedy approach that is described in [30]. The idea is to sort all edges $e \in \mathcal{E}$ in descending order according to their scores p_e being defined as:

$$p_e = p_{ij} = a_{ij} \cdot ||h_i - h_j||_2^2 \quad (6)$$

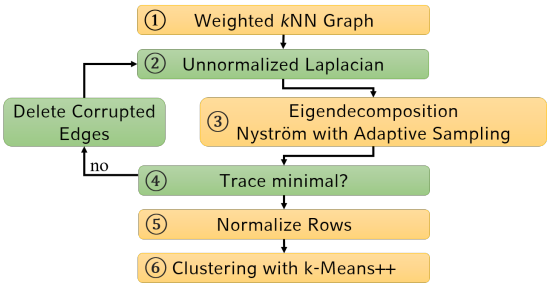


Figure 2: Overview of our method SCAR. Green boxes imply steps in our method that are analogue to RSC [7] and orange implies a significant change or addition.

We iteratively add edges to A^c such that the side constraints defined by parameters θ and m are fulfilled. Further details, proofs, and the reduction to the multidimensional knapsack problem [42] can be found in [7].

4 SCAR - SPECTRAL CLUSTERING ACCELERATED AND ROBUST

We propose our new clustering method SCAR (**S**pectral **C**lustering – **A**ccelerated and **R**obustified). SCAR separates the affinity graph of the data in an iterative approach into two latent components: a clean graph, which is used for the subsequent clustering, and a graph containing noisy edges. Likewise to Robust Spectral Clustering (RSC) [7], it detects noisy edges in each step that are disadvantageous for clustering. Therefore, it reaches overall robustness against noise compared to the original spectral clustering [40]. SCAR is significantly faster than RSC as we accelerate the most time-intensive step, the eigendecomposition, using the Nyström method [16] explained in Sec. 3.2. Furthermore, we improve several aspects of RSC significantly, such that SCAR is not only faster but also achieves significantly better results in real-world experiments as shown in Sec. 5. Fig. 2 gives an overview of our method SCAR and highlights the most important steps that deviate from RSC. In the following, we describe each step of our method in more detail. Algorithm 1 shows the corresponding pseudocode.

Step 1. We calculate the symmetric, weighted kNN graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ of the input data $\mathcal{X}$ (cf. line 1 in the pseudo-code). Each data point $x_i \in \mathcal{X}$ implies a node $v_i \in \mathcal{V}$, where $|\mathcal{X}| = |\mathcal{V}| = n$, i.e., there exists a bijective mapping $\phi : \mathcal{X} \to \mathcal{V}$. Further, $\mathcal{E} = \{(v_i, v_j) \mid \forall v_i, v_j \in \mathcal{V}, i \neq j : v_i \in kNN(v_j) \vee v_j \in kNN(v_i)\}$, and the weight of edges is defined by the Gaussian similarity function:

$$w(v_i, v_j) = exp\left(\frac{-||\phi^{-1}(v_i) - \phi^{-1}(v_j)||^2}{2\sigma^2}\right) \quad (7)$$

In our experiments, we use $\sigma = \sqrt{nd/2}$ per default. kNN graphs are suitable to find clusters of arbitrary shapes and varying density. Thus, when using spectral clustering, they are on most real world datasets superior to fully connected graphs (FCG), ϵ-neighborhood graphs, or Gabriel Graphs [25]. Our evaluation in Sec. 5 supports these findings. In contrast to RSC [7], we use a weighted kNN graph and apply the Gaussian kernel to weight the edges, which gives

more weight to closer points than to points farther away. This further improves clustering results in general [40], as it offers more information that can be relevant for clustering.

Step 2. As elaborated in [22], the unnormalized Laplacian is more suitable than normalized versions to discern between clusters and outliers resp. noise in the eigenspace, amplifying the difference between corrupted and clean edges later. Thus, we calculate the unnormalized graph Laplacian (line 4) as $L = D - A$ based on $\mathcal{G}$.

Step 3. We approximate the eigenvalues of L with the Nyström method as described in Sec. 3.2. Following [38], we use an adaptive sampling approach, where we choose $\alpha \cdot n$ points with the highest degrees as landmarks (cf. line 6). As noise points are unlikely to be nearest neighbors of nodes outside of their own neighborhood (compare to, e.g., ideas of outlier detection algorithms ODIN [19] or kNN-LOF [55]), their corresponding nodes in the kNN graph tend to have lower degrees. As we sample the nodes with the highest degrees, potential losses regarding the set of edges concern prevalently the noisy edges, we want to remove anyway. We then approximate the first k eigenvectors on $L_1 \in \mathbb{R}^{(\alpha n)\times(\alpha n)}$, which is a small submatrix of $L \in \mathbb{R}^{n\times n}$ (see Equation 1). The resulting matrix $\tilde{H}_1 \in \mathbb{R}^{\alpha n\times k}$ contains the first k approximated eigenvectors of L_1. The sampling of the submatrix is outlined in lines 7-10, whereas line 11 shows the eigendecomposition. In line 12 the Nyström extension is carried out. In Sec. 5 we thoroughly investigate several decomposition methods for computing the eigenpairs of L_1. In the following we work on $\tilde{H} \in \mathbb{R}^{n\times k}$, as obtained by Eq. 3.

Step 4. We check in line 15 of the pseudo-code whether the trace of $\tilde{H}^T L(A^g)\tilde{H}$ has decreased compared to the previous iteration. The identification and extraction of corrupted edges in the graph is described in Sec. 3.3 and follows the approach of RSC [7]. We apply Equation 6 in line 19, i.e., p_{ij} is calculated for all edges $(v_i, v_j) \in \mathcal{E}$. High values for p_{ij} indicate a high dissimilarity between the embeddings of nodes v_i and v_j, even though the nodes are connected by an edge. Thus, assigning an edge (v_i, v_j) with a high value p_{ij} to the noise component A^c improves the clustering quality as the edge is disregarded in the subsequent clustering step.

However, to ensure sparsity thresholds, bounds set with θ and m are respected [7]. The parameter θ prevents eliminating too many edges required for reasonable clustering results by limiting the maximum number of overall removable edges. The parameter m ensures a maximum local sparsity, i.e., each node keeps at least a portion m of its originally connected edges. In our experiments, we use $m = 0.5$ per default. If updates on the graph separation still lead to quality improvements, we recalculate L and approximate its eigendecomposition again with the Nyström method. We alternatingly update A^g in line 22 and $\tilde{H}$ until the trace cannot be significantly lowered.

Step 5. As suggested by [7, 11, 16], we orthogonalize and norm the first k resulting approximated eigenvectors row-wise (cf. line 24) which increases clustering quality and stability:

$$\bar{H}_{[i,:]} = \frac{\tilde{H}_{[i,:]}}{\|\tilde{H}_{[i,:]}\|_2} \tag{8}$$

Algorithm 1: SCAR Algorithm

Input: Dataset X, user input k, nn, α, θ, m
Output: Clustering containing assigned labels

1 $A \leftarrow$ kNN_graph(X, nn);
2 $A^g \leftarrow A$;
3 **for** $iter < max\_iterations$ **do**
4 $\quad L \leftarrow$ Laplacian(A^g) ; `// see Sec. 3.1`
5 $\quad$ `/* Nyström method */`
6 $\quad X_l \leftarrow \alpha \cdot |X|$ landmarks;
7 $\quad i \leftarrow$ indices_of(X_l);
8 $\quad j \leftarrow$ indices_of($X \backslash X_l$);
9 $\quad L_1 \leftarrow L[i, i]$;
10 $\quad L_2 \leftarrow L[j, i]$;
11 $\quad \tilde{H}_1, \Lambda \leftarrow$ eigendecomposition(L_1);
12 $\quad \tilde{H}_2 \leftarrow L_2 \tilde{H}_1 \Lambda^{-1}$; `// see Equation 3`
13 $\quad \tilde{H} \leftarrow$ reassemble($\tilde{H}_1, \tilde{H}_2$);
14 $\quad trace \leftarrow$ sum(Λ);
15 $\quad$ **if** *trace is minimal* **then**
16 $\quad\quad$ break;
17 $\quad$ **end if**
18 $\quad$ `/* Removing corrupted edges */`
19 $\quad p_{i,j} \leftarrow a_{i,j} \cdot \|h_i - h_j\|_2^2$; `// see Equation 6`
20 $\quad removed\_edges \leftarrow$ edges(argmax(p), θ, m);
21 $\quad A^c \leftarrow$ matrix($removed\_edges$);
22 $\quad A^g \leftarrow A - A^c$;
23 **end for**
24 $\bar{H} \leftarrow$ row-wise_norm($\tilde{H}$);
25 $Clustering \leftarrow k$_means++(rows($\bar{H}$));

Step 6. In the last step (shown in line 25), we cluster the first k rows of $\bar{H}$ (that has the eigenvectors of A^g as columns) with k-Means++ [2]. k-Means++ improves the selection of initial cluster centers for k-Means, leading to an earlier convergence and thus further speed-up compared to traditional spectral clustering approaches using k-Means.

5 EVALUATION

In the following, we examine our method SCAR thoroughly. In Sec. 5.1 we present our experimental setup. In Sections 5.2 and 5.3 we analyze SCAR's clustering quality, noise robustness, efficiency and scalability. In Sec. 5.4 we summarize SCAR's clustering and speed performance and regard their mutual dependencies. In Sec. 5.5 we evaluate the improvements of SCAR over RSC and SC. In Sec. 5.6 we evaluate the influence of various hyperparameters and design choices. SCAR retained an excellent balance between speed and quality over all experiments, while we refrained from hiding experiments that did not deliver desirable results in order to prevent overoptimism [8].

5.1 Experimental Setup

Datasets. In our evaluation, we use two synthetic datasets and ten real-world benchmark dataset. Both synthetic datasets, moons and

Table 1: Dataset properties used in the analysis.

	dataset	n	d	k	noise [%][9]	LB-UB [%][10]
syn.	moons	1,000	2	2	15	
	circles	1,000	2	2	15	
real	iris	150	4	3	7	5-9
	dermatology	366	33	6	9	4-14
	banknote	1,372	4	2	2	0-4
	pendigits_{16}	1,499	16	2	1	0-2
	pendigits_{146}	2,279	16	3	1	0-2
	pendigits	7,494	16	10	9	2-13
	USPS	11,000	256	10	24	12-33
	MNIST-10K	10,000	784	10	24	13-29
	MNIST-20K	20,000	784	10	21	11-27
	letters	20,000	17	26	46	20-61

circles, are constructed using data generator functions from the scikit-learn library.

Real world benchmark datasets *iris*, *dermatology*, *banknote*, *pendigits*, and *Letter Recognition* (*letters* for short) were obtained from the *UCI- MLR*[1]. *MNIST* and *USPS* were obtained from the repository of the *CS NYU*[2]. Similar to the work of [7], random subsamples were selected for the MNIST dataset. For the pendigits dataset, specific subsets pendigits_{16} and pendigits_{146} were defined as benchmark datasets in the literature [7, 23, 35]. For *dermatology* we omit the feature about the *age* of patients as the dataset is incomplete w.r.t this feature. The data statistics are summarized in Tab. 1.

Competitors. We compare SCAR with standard Spectral Clustering (SC) [40][3], Robust Spectral Clustering (RSC) [7][4], normalized SpectACl [21][5], and Density Core Finding (DCF) [50][6].

Implementation Details. SCAR is implemented in Python, building off of the implementation of RSC [7][6]. We additionally use the libraries scikit-learn, NumPy, Scipy and slepc4py/petsc4py [7]. Experiments were run on an Intel(R) Xeon(R) Silver 4208 CPU @ 2.10GHz using 32GB RAM.

Code: available on GitHub [8]

Hyperparameter Setting. For the synthetic datasets we use per default 0.15 for the parameter *noise* that regulates the jitter. Note that this value is significantly higher than the values applied in, e.g., RSC [7]. We tune $\alpha \in [0.1, 0.2, \ldots, 0.9]$. For every dataset, we fix the parameter θ in all our experiments dataset-specific, where $\theta \in \{20, 30, 200, 500, 1k, 10k, 30k, 60k\}$. Following the rule-of-thumb popularized by [14], we used $2\sqrt{n}$ as an upper bound for *nn* and tested values in 10 percent steps for all methods, accordingly. For a fair comparison to the competitor *DCF*, we also evaluated the parameter β used in their method in the range of $[0.1, 0.2, \ldots, 0.9]$ to obtain best scores for the cluster metric.

[1]https://archive.ics.uci.edu/ml/index.php (retrieved: Feb 18, 2022)
[2]https://cs.nyu.edu/~roweis/data.html (retrieved: Feb 25, 2021)
[3]https://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html (last accessed: Jul 14, 2022)
[4]https://github.com/abojchevski/rsc(last accessed: Jul 14, 2022)
[5]https://bitbucket.org/Sibylse/spectacl/src/master/ (last accessed: Jul 14, 2022)
[6]https://github.com/tobinjo96/DCFcluster (last accessed: Jul 14, 2022)
[7]https://slepc.upv.es/documentation/ (last accessed: Jul 14, 2022)
[8]https://github.com/SpectralClusteringAcceleratedRobust/SCAR.git

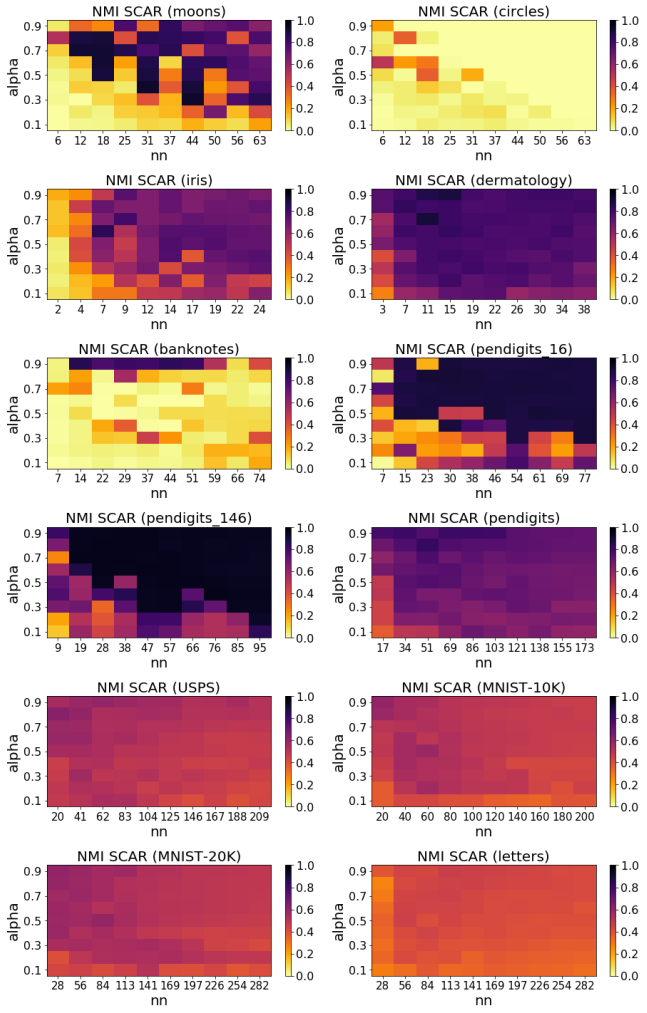


Figure 3: Summary of NMI obtained with SCAR depending on *nn* and α for all datasets.

5.2 Clustering Quality

Clustering quality is measured using the *Normalized Mutual Information* (NMI) [48] and *Adjusted Rand Index (ARI)* scores, which range from 0 to 1. Higher values imply a better accordance to the ground truth. Following the suggestion of [44], ARI should be used when the reference clustering has large equal sized clusters; scores based on mutual information should be used when the reference clustering is unbalanced and there exist small clusters. In the following, we run all experiments for 10 trials and report the average clustering scores per parameter setting if not stated otherwise.

5.2.1 *Effectiveness.*

In Tab. 2 on the left, we summarize the best NMI and ARI scores evaluated on each dataset. In order to obtain the best outcomes for

[9]for synth. datasets noise is added in the sklearn function as standard deviation of Gaussian noise, for real datasets, noise is measured as ratio of inter-cluster edges vs. total edges in the *k*-Graph for all tested *nn*
[10]LB = noise lower bound, UB = noise upper bound

Table 2: Maximum clustering quality reached, measured by normalized mutual information (NMI) scores and adjusted rand index (ARI), as well as minimum runtimes (in seconds) reached for best NMI scores and overall. Best/Second-best results are **bold**/underlined. Values regarded closer in the text are marked in red for faster readability.

		max NMI ARI										min runtime of best NMI (min runtime overall)				
	dataset	SC		RSC		DCF		SpectACl		SCAR		SC	RSC	DCF	SpectACl	SCAR
syn.	moons	0.43	0.72	0.43	0.72	0.91	0.96	0.88	**0.98**	**0.92**	0.96	0.15 (0.11)	0.19 (0.14)	0.14 (0.13)	0.11 (0.08)	**0.06** (**0.03**)
	circles	0.00	0.00	0.19	0.08	0.33	0.16	**0.79**	**0.86**	0.50	0.57	0.13 (0.11)	0.32 (0.20)	0.09 (0.07)	0.07 (0.06)	**0.05** (**0.03**)
real	iris	0.82	0.83	0.81	0.75	0.77	0.75	0.76	0.73	**0.84**	**0.85**	**0.03** (**0.02**)	0.04 (0.04)	0.08 (0.06)	0.06 (0.04)	**0.03** (**0.02**)
	dermatology	**0.93**	0.91	**0.93**	**0.92**	0.91	0.88	0.88	0.88	0.89	0.89	**0.03** (**0.03**)	0.09 (0.05)	0.09 (0.08)	0.08 (0.08)	0.05 (0.04)
	banknote	0.61	0.62	0.61	0.62	0.62	0.62	0.02	0.03	**0.86**	**0.90**	0.16 (0.15)	0.35 (0.19)	0.11 (0.09)	**0.10** (0.10)	0.12 (**0.03**)
	pendigits$_{16}$	**0.92**	**0.95**	0.90	0.94	0.78	0.76	0.22	0.10	0.90	0.94	0.26 (0.18)	0.37 (0.21)	**0.13** (0.12)	0.17 (0.14)	**0.13** (**0.08**)
	pendigits$_{146}$	0.95	0.96	**0.96**	**0.97**	0.87	0.86	0.70	0.58	0.95	**0.97**	0.41 (0.41)	0.87 (0.69)	0.29 (0.26)	0.29 (0.29)	**0.27** (**0.17**)
	pendigits	0.81	0.67	0.82	0.67	**0.84**	**0.76**	0.74	0.59	0.82	**0.76**	3.88 (2.94)	8.25 (4.05)	**0.96** (**0.80**)	2.09 (1.73)	2.68 (1.38)
	USPS	0.65	0.46	**0.68**	0.45	0.60	0.31	0.58	0.42	0.63	**0.48**	22.22 (22.22)	10.33 (9.70)	55.42 (54.89)	**4.00** (3.86)	4.59 (**3.18**)
	MNIST-10K	0.67	0.50	**0.74**	**0.55**	0.59	0.45	0.62	0.50	0.61	0.44	36.29 (36.29)	10.49 (10.49)	114.03 (111.82)	**5.00** (4.91)	7.34 (**4.41**)
	MNIST-20K	0.68	0.51	**0.76**	**0.55**	0.62	0.49	0.63	0.49	0.60	0.52	244.87 (244.87)	46.45 (31.39)	444.92 (385.94)	**21.18** (**21.18**)	38.83 (**21.18**)
	letters	0.42	0.16	0.42	0.13	**0.56**	0.17	0.38	0.12	0.46	**0.22**	418.02 (62.48)	38.29 (38.29)	**8.94** (**8.91**)	13.88 (12.99)	19.06 (10.84)
	Avg.	0.65	0.60	0.68	0.61	0.70	0.59	0.6	0.52	**0.74**	**0.70**	60.53 (30.81)	9.67 (7.95)	52.1 (46.93)	**3.91** (3.78)	6.10 (**3.44**)

each dataset and each method, we applied a grid-search over the respective parameter spaces as outlined in Sec. 5.1. The dependence of NMI's on the number of neighbors *nn* can be seen in Fig. 4. We discuss the runtimes shown in the right part of Tab. 2 – also in combination with the quality of the clusterings – in Sec. 5.3 and analyze influence of hyperparameter settings in Sec. 5.6.

SCAR reaches on average the best NMI/ARI scores while those results were reached on average with the second best runtime of all tested algorithms. SCARs average runtime is approximately an order of magnitude faster compared to the original standard spectral clustering algorithm (SC) and DCF. SCAR always returns clusterings of solid quality, in contrast to, e.g., SpectACl, which is not able to find an acceptable clustering for the datasets banknote or subsets of the pendigits dataset (marked in red in Tab. 2, see also Fig. 4). Second best values were often reached by SC, which is, however, not designed to reach fast runtimes. SC's good results on our benchmark datasets confirm the high potential of spectral methods for high-quality clustering results. Further, we observe that SCAR can handle highly noisy datasets like moons, where SC as well as RSC could not correctly detect the clusters, reaching NMIs (ARIs) of only 0.43 (0.72). We regard the sensitivity of all methods w.r.t. the parameter *nn* in Fig. 4. Where most methods are rather robust w.r.t. the parameter *nn*, their default settings may not be optimal: in Fig. 1, we applied all algorithms' default parameter settings on the moons dataset. Here, none of our competitors could find the clusters correctly. In contrast, we optimized parameter settings w.r.t. the NMI/ARI via a grid search for Tab. 2. Furthermore, we perceived the banknote dataset as an interesting case, as SCAR significantly surpassed its competitors. SpectACl, e.g., was not able to produce any meaningful clustering results over a variety of tested parameter settings, reaching a maximum NMI (ARI) of 0.02 (0.03). All other competitors reached NMI/ARI scores around 0.62. The banknote dataset contains 4-dimensional representations of forged and authentic banknotes. Its clusters overlap in all dimensions, making its similarity graph highly noisy. Thus, the advantages of SCAR's noise robustness can be seen here, yielding an outstanding NMI (ARI) of **0.86** (**0.90**) for our method.

Even though SCAR yields very good results for most datasets, we still see room to further improve clustering results on high-dimensional datasets in future work. Especially, performance on datasets emerging from pixel-data (USPS and MNIST versions) could benefit from applying feature weighting approaches as outlined in Sec. 2. Tab. 2 shows that despite their different strengths, the clustering metrics do not differ much in how the investigated methods compare. Thus, only NMI is reported in the following as the default metric.

5.2.2 *Robustness against Noise.*

To evaluate SCAR's robustness against noise, we fix the parameter settings for *nn* and α, and only modify the amount of *jitter* in the range of $[0.0, 0.05, 0.1, \ldots, 0.03]$ on the moons dataset. The left graph in Fig. 5 shows that SCAR consistently outperforms other models on the moons dataset for high noise levels ($noise > 0.2$). The NMIs of most comparative methods drop heavily for noise values over 0.1, resp., 0.2. Qualitative results can also be seen in Fig. 1, where SCAR is the only method able to correctly discern the two moons for a comparably high noise level of $noise = 0.15$. The right graph in Fig. 5 gives all runtimes in seconds. SCAR shows an almost constant runtime over different levels for *noise*. For RSC we observe higher runtimes as the eigendecomposition on the whole Laplacian is computed in each iteration. Notably, DCF also shows increased runtimes for low noise values due to higher densities within the clusters. The efficiency of our model evaluated on different benchmark datasets is further discussed in the next section.

Similar to [7], we also examine the robustness against *noisy edges*: We generated Gaussian distributed clusters (blobs) and versions of the moons datasets where we added "corrupted" edges to the associated kNN graph. I.e., we added edges between nodes of different clusters using the planted-partition model. Intra-cluster edges were created with a probability of 30% and we added noise edges s.t. 10%, resp., 20% of all edges in the kNN graph were corrupted. We evaluate the precision $p = |\mathcal{E}_c \cap \mathcal{E}_r|/|\mathcal{E}_c|$ and recall $r = |\mathcal{E}_c \cap \mathcal{E}_r|/|\mathcal{E}_r|$, where $\mathcal{E}_c$ is the set of corrupted edges and $\mathcal{E}_r$ is the set of edges removed by SCAR. In contrast to [7], we also regard the effect of removing corrupted edges on the clustering

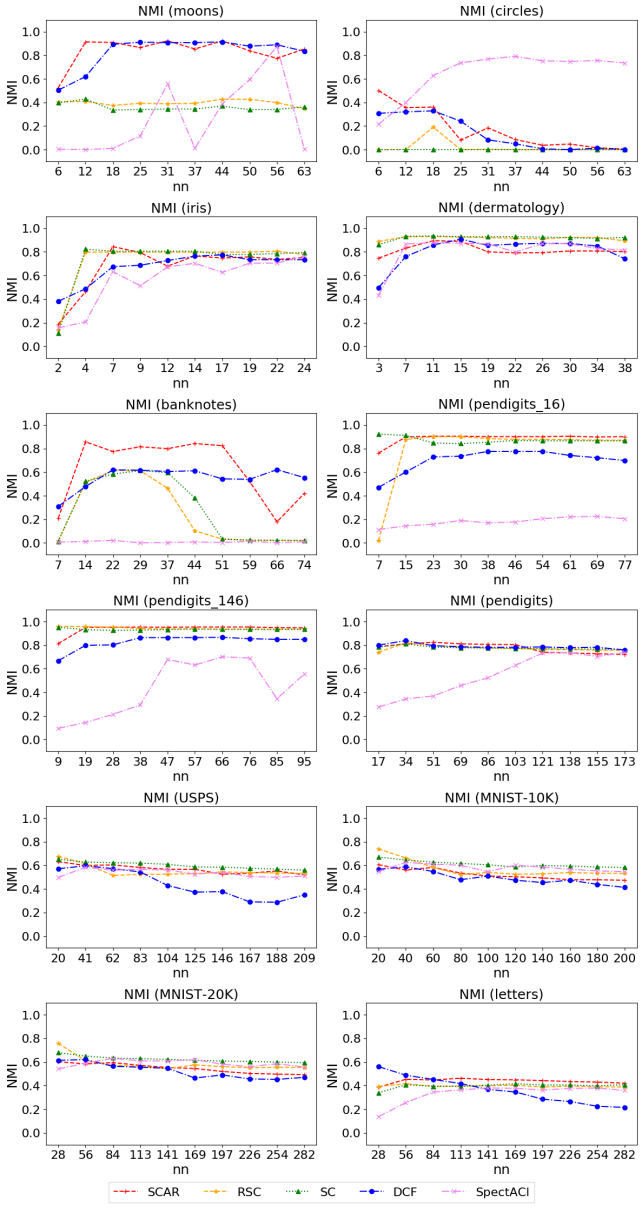


Figure 4: Comparison of NMI of all methods with their best parameter settings on all datasets depending on *nn*.

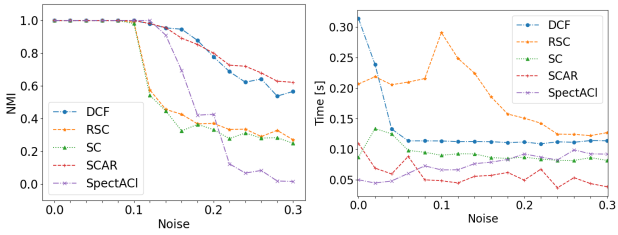
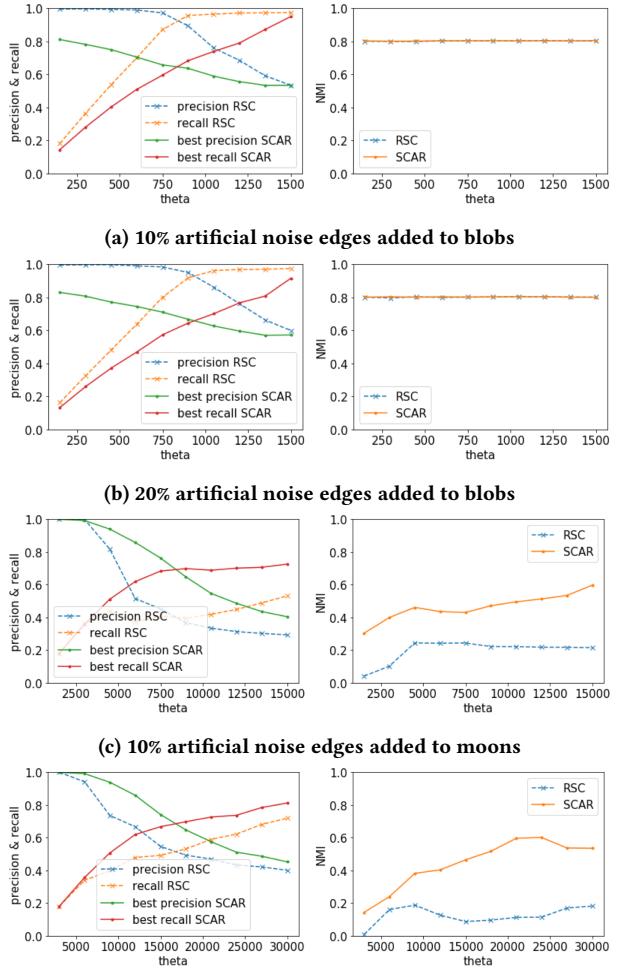


Figure 5: NMI scores (left) and runtime in [s] (right) for $noise \in [0, \ldots, 0.3]$ in 0.02 steps on moons dataset.



(a) 10% artificial noise edges added to blobs

(b) 20% artificial noise edges added to blobs

(c) 10% artificial noise edges added to moons

(d) 20% artificial noise edges added to moons

Figure 6: Precision and recall (left) and NMI (right) for 10% or 20% artificial noise edges added to blobs (n=1000, k=20) resp. moons averaged over random_state=$[0-9]$.

quality.Figures 6a and 6b show precision and recall of the detected corrupted edges, as well as the achieved NMIs of RSC and SCAR for increasing θ. Even though precision and recall – implying the quality of detecting corrupted edges – of SCAR's results are lower than for RSC, this does not affect the overall clustering quality. Instead, the constant NMI scores, while increasing θ for both cases (10% and 20% added noise edges), indicate that corrupted edges do not affect the obtained clustering quality for Gaussian distributed clusters. Figures 6c and 6d imply that this is different for moons datasets. Here, SCAR surpasses RSC w.r.t. precision and recall on both noise settings throughout almost all tested values for θ. In contrast to the Gaussian distributed clusters, for the moons dataset, removing corrupted edges enables a higher clustering quality: The NMIs of SCAR are significantly higher than the NMIs for RSC throughout all tested values for θ. Fig. 6 shows that SCAR surpasses RSC in the

detection of corrupted edges exactly where these corrupted edges impede a high quality clustering by connecting hard-to-distinguish clusters.

5.3 Runtime Analysis

In this section, we evaluate our model's runtime. In Sec. 5.3.1, we provide a general overview of SCAR's efficiency compared to state-of-the-art methods. We perform scalability experiments in Sec. 5.3.2 and regard the complexity in Sec. 5.3.3

5.3.1 *Efficiency.*

Tab. 2 shows on the right the minimum runtime in seconds for the trials with the highest NMI scores as well as the minimum runtime of all tested parameter settings in brackets. We observe that SCAR yields its best results w.r.t. the NMI almost always as the fastest or second fastest algorithm. SCAR can generally provide faster results than standard SC. The speed-up, in particular, increases with larger datasets. Only on the quite small dermatology dataset, SCAR runs 0.02 seconds longer than SC. The highest speed-up is reached on the letters dataset, where SCAR is more than 20 times faster than SC, while simultaneously increasing the NMI by 10%. Using a similar design and notion of noise as RSC, it is noteworthy that we surpass RSC w.r.t the runtime on every tested dataset. Note also, that RSC already accelerates the eigendecomposition by leveraging IRLM. We reach a maximum acceleration factor of 6.4 in relation to RSC for the heavily noisy circles dataset, where we simultaneously improve the clustering quality from an NMI (ARI) of 0.19 (0.08) to 0.50 (0.57). For further runtime comparisons with RSC and SC, see Sec. 5.5.2. Even though DCF shows fast runtimes as well as good clustering results for most datasets, it cannot reach acceptable runtimes on high-dimensional datasets like USPS or MNIST variations (marked in red). Whereas experiments on lower-dimensional datasets show comparable runtimes in the same order of magnitude for all algorithms, DCF needs on these three high-dimensional datasets more than ten times longer than SCAR. Further investigations on the dependence of all algorithm's runtimes can be found in Sec. 5.3.2. SpectACl shows, similar to SCAR, good runtimes for its best results, but mostly returns significantly worse clustering results. Especially SpectACl's performance on the datasets banknote and the first two pendigits versions (marked in red) is of surprisingly low quality.

For some parameter settings, algorithms may have significantly lower runtimes than for others. E.g., for a small number of nearest neighbors *nn*, the respective nearest neighbors graph has less edges, and thus, most operations performed on it are faster. Analyzing the values in brackets in Tab. 2, we see the best runtimes over all tested parameter settings that can be reached for each experiment and each algorithm. I.e., in contrast to the runtimes regarded in the last paragraph, where we optimized parameter settings for a high NMI, we now optimize parameter settings for a low runtime. Also here, SCAR reaches most often the fastest runtimes. We note, that even for the most suitable parameter settings, DCF cannot achieve an acceptable runtime on high-dimensional datasets like USPS and MNIST variations (marked in red).

We observed that for all datasets, the minimum runtime for the best NMI results were usually close to the respective minimum runtime over all tested settings. More precisely, most of them were at maximum twice as high as the fastest runtime for the respective

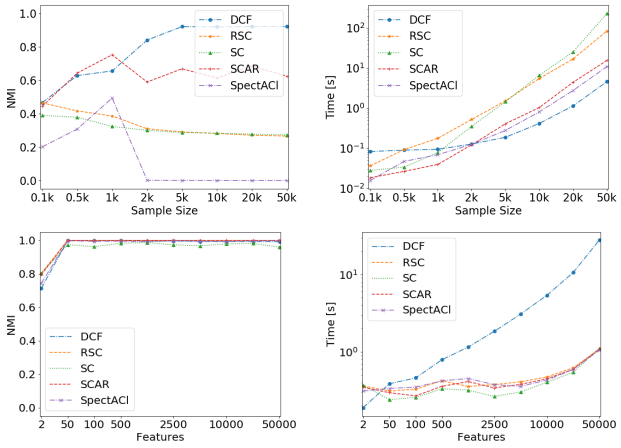


Figure 7: NMI scores (left) and runtime in seconds (right). Top: Moons dataset (noise=0.15) with varying n. Bottom: Blobs dataset (n=1000, k=20, random_state=None) with varying d.

algorithm. This supports the idea that the model has a stronger influence on its runtime than the selected parameter setting.

In conclusion, we found that SCAR is stable in its anticipated clustering quality and yields good results at high speed.

5.3.2 *Scalability on Synthetic Datasets.*

Fig. 7 shows the scalability of our approach on the moons and blobs datasets, with a fixed noise of 0.15 for moons and a default cluster standard deviation of 1.0 for blobs. On the former, we scale the number of data points in the range $[100, \ldots, 50k]$. On the latter, we scale the number of features $[2, \ldots, 50k]$. The number of neighbors that are taken into account for the construction of the kNN graph is set to $nn = \sqrt{n}$ in both experiments, where n denotes the number of data samples. All other parameters are frozen. The left diagrams show the obtained NMI scores, and the right diagrams show the elapsed time for all evaluated methods. On the moons dataset, SCAR's scalability oupterforms RSC and SC w.r.t. both, computational time as well as obtained NMI scores for increasing sample size. For smaller datasets, our approach also shows superior performance compared to DCF, which cannot be maintained for increasing the sample-size. However, DCF's runtimes deteriorate for higher dimensionalities, as can be seen in the lower part of Fig. 7 (note the log-scale). SCAR's runtime stays almost linear in the number of features. In Fig. 8, DCF's unfortunate runtime behavior w.r.t. the dimensionality can also be observed on larger real-world datasets with higher dimensionality. While DCF yields low runtimes for large datasets if they are low-dimensional, e.g., for letters, its runtime tremendously increases on USPS, MNIST-10K and MNIST-20k.

5.3.3 *Complexity Analysis.*

Having the same fundamental structure as RSC, we refer for our complexity analysis on the explanations of [7], showing a runtime approximately linear in the number of edges. In the following, we elaborate on the differences between SCAR and RSC that potentially influence the complexity (see also Fig. 2). In Step (1), we calculate the weighted kNN graph in contrast to the unweighted kNN graph

for RSC and apply a Gaussian kernel on the edge weights. These changes do not increase the runtime complexity, as all edges of the kNN graph are accessed in both approaches. In Step (3), [7] use the power iteration for the eigendecomposition. We reduce the runtime by using the Nyström method, see Sec. 3.2. In Step (5), we normalize the rows of the approximated, cleansed matrix $\tilde{H} \in \mathbb{R}^{n \times k}$, where $\tilde{H}$ contains the first k vectors that are needed for the subsequent clustering step. Usually, we have $k \ll \sqrt{n}$, s.t. the complexity is not increased when working on a kNN graph (containing approximately $O(n\sqrt{n})$ edges [39]). Overall, we reach a similar complexity as RSC, which is approximately linear in the number of edges, while improving the runtime. Our experiments in Sec. 5.3 confirm the improved runtime w.r.t. RSC.

5.4 Effectiveness and Efficiency

Fig. 8 summarizes the models' performances on the various datasets, where the x-axis shows the runtime and the y-axis shows the clustering scores. Optimal results are located in the upper left reflecting a high NMI score reached within a short amount of time. We show only the best runs of all methods to reduce visual clutter, i.e., only runs that yielded at least 75% of the best NMI score reached by the respective method are shown as single dots. On the highly noisy moons dataset, SCAR's robustness and efficiency dominates the other methods in terms of both, clustering performance and runtime. On small real-world datasets (iris, dermatology, banknotes and the pendigits variations), SCAR is highly competitive with other state-of-the-art models w.r.t. NMI and runtime. As all tested methods have runtimes under one second for all smaller datasets, larger datasets are more expressive for runtime analyses. Thus, we regard in the following (as well as in Fig. 10) the datasets with more than 5000 points (pendigits, USPS, MNIST versions and letters) when investigating runtimes. We note that SCAR is comparably fast on these datasets *and* reaches low runtimes with a comparably low variance. For the low-dimensional datasets pendigits and letters, DCF is even faster than SCAR, but for higher-dimensional datasets (USPS and MNIST versions) advantages of using any of the newer spectral clustering approaches become clear, as DCF's runtime does not scale with the dimensionality. In summary, Fig. 8 demonstrates that SCAR nearly always outperforms its competitors in either runtime, clustering quality, or both and particularly highlights SCAR's reliability.

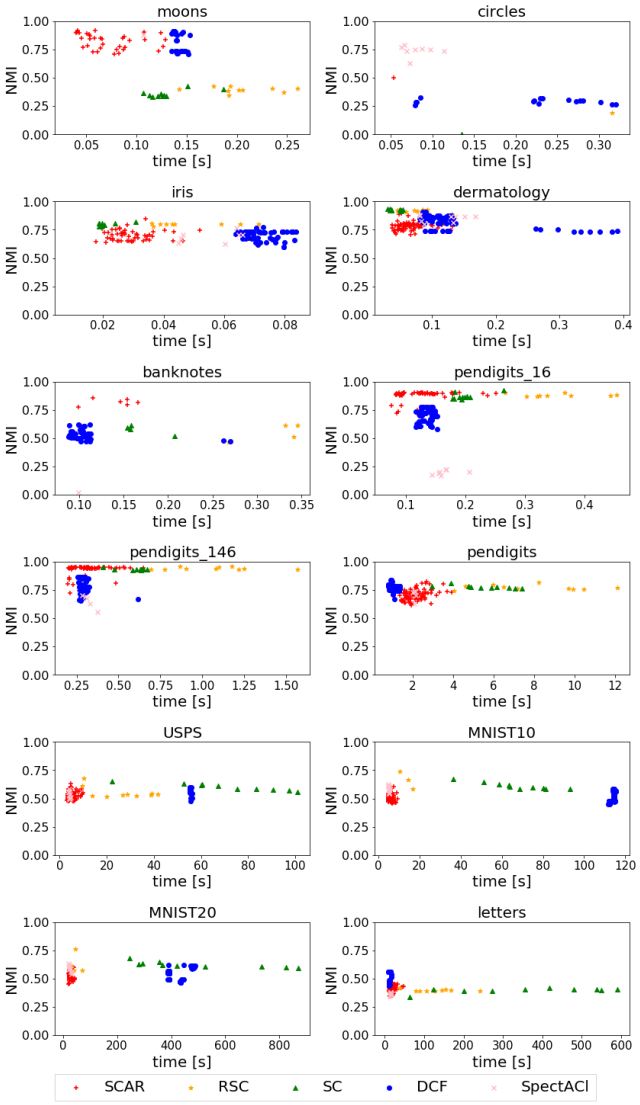


Figure 8: Runtimes and NMIs of all experiments that reached at least 75% of each method's respective best NMI.

5.5 Improvements over RSC and SC

In the following, we examine the improvements of SCAR over RSC and the original Spectral Clustering algorithm (SC) in more detail. Sec. 5.5.1 regards the single components that differentiate SCAR from RSC as well as their functional interaction. Sec. 5.5.2 regards runtime improvements over RSC.

5.5.1 Effectiveness Improvements of SCAR over RSC and SC. Fig. 9 shows NMIs on the highly noisy moons dataset for various settings for *nn* and different methods: on the left, we compare RSC with a straight-forward Nyström-accelerated version of RSC and our method SCAR. On the right, we perform an ablation study w.r.t. the changes between RSC and SCAR. (We condense the results by setting α to our recommended default value $\alpha = 0.7$). The left part of the figure shows that a simple speed-up of RSC would lead to significantly worse results, whereas SCAR drastically improves the results of RSC. On the right, we can see that each of SCAR's components is chosen meaningfully, leading to an improvement of quality that is reached by the elaborated combination of concepts rather than any single adaption. We regarded the reasons for the individual components in Sec. 4 and explain their impacts and synergies in the following. Using an unweighted graph can deliver good results on the moons dataset, if exactly the right number for *nn* is chosen (i.e., such that only very few corrupted edges exist). However, as seen in the first line of Fig. 9 on the right, this leads to a strong and unpredictable dependence on guessing a good value for *nn*. Weighting the edges also allows for a more meaningful sampling of the edges for the Nyström method with the adjusted sampling method we apply: as corrupted edges connect nodes of different

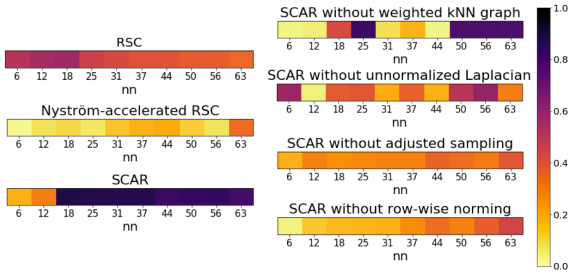


Figure 9: Ablation study of SCAR's NMI performance on moons (avg. over 10 random instantiations) depending on nn and for fixed $\alpha = 0.7$. Dark colors imply better NMI scores.

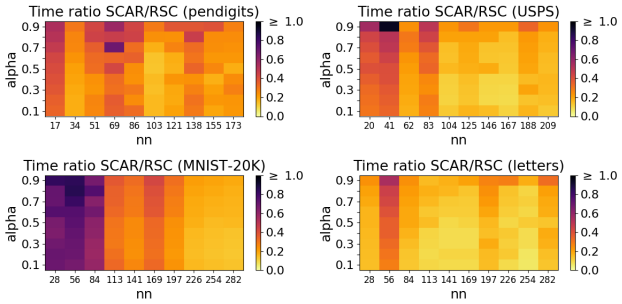


Figure 10: Summary of time ratios SCAR / RSC depending on nn and α for large real world datasets. Lighter colors imply better (i.e., faster) results for our method SCAR.

clusters (and distances between clusters are larger than distances inside clusters) they tend to be longer than non-corrupted edges. Applying the Gaussian kernel for calculating the edges' weights, this leads to smaller degrees of nodes connected by corrupted edges. Thus, sampling the nodes with higher degrees allows to sort out corrupted edges (compare with line 3 on the right of Fig. 9). Using an unnormalized Laplacian further enhances the distinguishability between corrupted and non-corrupted edges [22] in the eigenspace, reinforcing the positive effects of our adjusted sampling method heavily (see line 2 on the right of Fig. 9). Small perturbances in the data can be compensated by normalizing the rows [16]. That accounts for jitter and pushes points of a cluster even closer in eigenspace, which robustifies the adapted sampling step and further improves the clustering (line 4 on the right of Fig. 9).

5.5.2 Efficiency Improvements over RSC and SC.
Fig. 10 shows SCAR's substantial runtime accelerations over RSC depending on their parameter settings on four larger benchmark datasets (where $n \geq 5000$, resp., $runtime > 1s$). In Fig. 8, runtimes and their respective NMIs are shown for all methods. In particular for larger datasets, SCAR nearly always outperforms SC and RSC regarding runtime significantly. A more thorough discussion on the proper choice of hyperparameters α and nn is given in Sec. 5.6.1.

5.6 Hyperparameter Tuning

In this section we examine the influence of various parameter settings on our model's performance. In Sec. 5.6.1, we examine the portion α denoting the number of landmarks chosen for the Nyström subsample and the number of nearest neighbors nn for the construction of the kNN graph. In Sec. 5.6.2, we evaluate the performance of various decomposition methods on the sampled submatrix and how the clustering quality and computational time depends on different configurations. In Sec. 5.6.3, we investigate the influence of the parameter θ on the models' performances of SCAR and RSC.

5.6.1 Number of Landmarks and Number of Neighbors.
In Fig. 3, we show the NMI scores for all tested datasets depending on nn and α, where darker, resp. lighter, colors reflect higher, resp. lower, NMI scores. For a more thorough analysis of the impact of the number of neighbors, we use $2\sqrt{n}$ as an upper bound for nn [14]. We see that the choice for nn and α has a strong effect on the clustering quality: The quality of smaller datasets depends more heavily on a proper choice of α compared to larger datasets. Our experiments show that a higher amount of landmarks improves clustering results. Furthermore, the illustration reveals that on larger datasets, the performance is improved whenever the kNN graph retains its sparse nature, i.e., by lowering the amount of nn. This effect also heavily improves the efficiency of our proposed method as discussed in Sec. 5.3.1. On USPS, as well as on the MNIST datasets, we observe higher peaks for lower values of nn. On smaller datasets, it is more likely that the kNN graph connects samples from distinctive cluster, i.e., the graph contains misleading information. Comparing iris and dermatology, we found that for the latter, it is more favorable to choose a smaller nn to identify the six clusters properly, whereas on iris, with three clusters, we can choose higher values without mingling the information of separate clusters. Per default, we suggest to use values $\alpha = 0.7$ and $nn = \sqrt{n}$.

While good clustering results are a prerequisite for useful clustering algorithms, SCAR's major benefit is its runtime acceleration. Fig. 10 summarizes obtained runtime quotients of SCAR compared to RSC for four large real-world datasets and their dependence on nn and α. We only display the larger datasets here, as they require runtimes for RSC $\gg 1s$ (see Table 2), and therefore an acceleration analysis is more meaningful. The values in each heatmap depict the ratio of runtimes between SCAR and RSC, i.e., $runtime(SCAR)/runtime(RSC)$. Consequently, smaller values indicate faster runtimes of SCAR compared to RSC. The effect and strength of the Nyström method can be observed for all larger datasets. By sampling only a submatrix in order to approximate the spectrum as a whole, we observe a performance boost compared to RSC. The impact of the choice of α is shown on the y-axis, whereas the effect of nn is shown on the x-axis. The experiments show that SCAR has significantly lower runtimes than RSC even for high values of α, further supporting our quite high recommended choice for $\alpha = 0.7$. For larger values of nn SCAR's speed-up becomes even clearer: Larger nn lead to more edges in the kNN graph and therefore more acceleration potential for SCAR over RSC as the graph is more dense. Fig. 4 indicates that SCAR's clustering results are relatively robust against the choice of nn. Thus, SCAR's runtime improvements over RSC do not have a negative effect on its clustering performance.

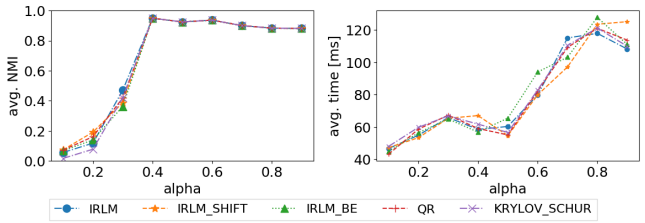
5.6.2 Decomposition of Submatrix.
In the following, we evaluate commonly used decomposition methods on the sampled submatrix of the Nyström Approximation explained in Sec. 3.2. Fig. 11 shows the highest observed NMI scores (left) within 10 trials as well as the respective runtimes (right) with a fixed value of *nn* for each dataset. As the submatrix in the Nyström method is symmetric, we apply the *Implicitly Restarted Lanczos Method* (IRLM) which is based on power iterations and has also been used in [7] as decomposition heuristic. Additionally, we evaluate variants of IRLM with *-Shift* applying a shift-inversion on the spectrum to transform the smallest eigenvalues to be the highest, and *-BE* for which eigenvalues are approximated from both ends of the spectrum. For the latter, [41] showed, that approximating eigenpairs from both ends of the spectrum can speed-up the convergence. We also evaluate a standard QR decomposition, as well as the *Krylov-Schur* decomposition as proposed by [47].[11] Empirically, all decomposition methods yielded similar qualitative results w.r.t the NMI score. Examining the runtimes on smaller datasets, we observe a slight overhead in the computation of the shifting operation for *IRLM-Shift*, as well as in applying a sampling from both ends of the spectrum. On larger datasets, this effect flattens out and the *Krylov-Schur* decomposition that is optimized towards large, sparse matrices shows a marginal benefit for larger α values. In our experiments we used the standard IRLM as default heuristic for the computation of the eigenpairs as it showed competitive results over the full range of the tested datasets.

5.6.3 Influence of Parameter θ.
In Fig. 12, we evaluate the influence of parameter θ on the clustering's quality and runtimes for SCAR and RSC [7]. As argued in Sec. 5.6.1, we fix *nn* to $nn = \sqrt{n}$. We scale the number of expected corruptions in the dataset logarithmically: $\theta \in [10, 100, 1k, 10k]$. On the moons dataset, our approach outperforms RSC almost over the full range of chosen θ whilst drastically reducing the computational time as shown on the right. Generally, increasing the sparsity threshold might lead to a clearer separation, however, the clustering quality suffers for very large values as clean edges might be attached to the corrupted graph A^c.

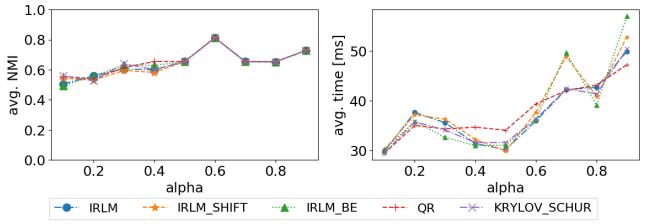
6 CONCLUSION

We introduced SCAR, a novel robust and efficient clustering method. It elucidates the benefits from Robust Spectral Clustering [7] enhanced by the Nyström method for an accelerated computation of the eigendecomposition. We reduced the sensitivity to noisy input data as well as the runtime complexity compared to standard Spectral Clustering significantly. In a thorough experimental study, we compare SCAR's clustering quality with state-of-the-art models showing highly competitive results on real-world benchmark datasets, as well as its robustness against noise on artificial data. We evaluated robustness w.r.t. noisy edges in the similarity graph of the data as well as robustness w.r.t. jitter in the original data, tackling the two most difficult types of noise for clustering. SCAR consistently yielded low runtimes, in particular it is significantly faster than RSC and SC, while returning highly competitive clustering qualities on real-world and synthetic data. SCAR is recommendable

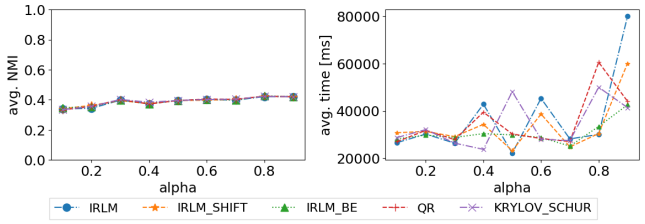
[11] We use state-of-the-art libraries, where IRLM and its variants are implemented in ARPACK, QR in LAPACK, and krylov-schur as part of SLEPc/PETSc



(a) synthetic data - moon



(b) real data - Iris



(c) real data - Letter Recognition

Figure 11: Avg. NMI scores (left) and runtimes (right) for decomposition methods IRLM, IRLM-Shift, IRLM-BE, QR and krylov-schur on different datasets.

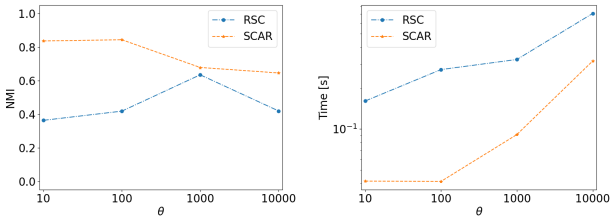


Figure 12: NMI scores (left) and runtime in [s] (right) for $\theta \in [10, 100, 1k, 10k]$ on moons dataset (noise=0.15).

when looking for a reliable, fast and robust clustering method on large and high-dimensional datasets that tend to be noisy.

ACKNOWLEDGMENTS

This work has been partially funded by VILLUM FONDEN, the Technical University of Munich's Institute for Ethics in Artificial Intelligence (IEAI) and the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A. The authors of this work take full responsibilities for its content.

REFERENCES

[1] Walter Edwin Arnoldi. 1951. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of applied mathematics* 9, 1 (1951), 17–29.

[2] David Arthur and Sergei Vassilvitskii. 2006. *k-means++: The advantages of careful seeding*. Technical Report. Stanford.

[3] Francis Bach and Michael Jordan. 2004. Learning spectral clustering. *Advances in neural information processing systems* 16, 2 (2004), 305–312.

[4] Sivaraman Balakrishnan, Min Xu, Akshay Krishnamurthy, and Aarti Singh. 2011. Noise thresholds for spectral clustering. *Advances in Neural Information Processing Systems* 24 (2011).

[5] Anna Beer, Ekaterina Allerborn, Valentin Hartmann, and Thomas Seidl. 2021. KISS-A fast kNN-based Importance Score for Subspaces. In *EDBT*. 391–396.

[6] Serge Belongie, Charless Fowlkes, Fan Chung, and Jitendra Malik. 2002. Spectral partitioning with indefinite kernels using the Nyström extension. In *European conference on computer vision*. Springer, 531–542.

[7] Aleksandar Bojchevski, Yves Matkovic, and Stephan Günnemann. 2017. Robust Spectral Clustering for Noisy Data: Modeling Sparse Corruptions Improves Latent Embeddings. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 737–746.

[8] Anne-Laure Boulesteix. 2015. Ten simple rules for reducing overoptimistic reporting in methodological computational research. *PLoS Computational Biology* 11, 4 (2015), e1004191.

[9] Eamonn Cahill, Alan Irving, Christopher Johnston, James Sexton, Ukqcd Collaboration, et al. 2000. Numerical stability of Lanczos methods. *Nuclear Physics B-Proceedings Supplements* 83 (2000), 825–827.

[10] Xiaojun Chen, Weijun Hong, Feiping Nie, Dan He, Min Yang, and Joshua Zhexue Huang. 2018. Spectral clustering of large-scale data by directly solving normalized cut. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1206–1215.

[11] Anna Choromanska, Tony Jebara, Hyungtae Kim, Mahesh Mohan, and Claire Monteleoni. 2013. Fast Spectral Clustering via the Nyström Method. In *International Conference on Algorithmic Learning Theory*. Springer, 367–381.

[12] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. 2004. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 551–556.

[13] Petros Drineas and Michael W Mahoney. 2005. Approximating a gram matrix for improved kernel-based learning. In *International Conference on Computational Learning Theory*. Springer, 323–337.

[14] Richard O. Duda, Peter E. Hart, and David G. Stork. 2001. *Pattern Classification* (2 ed.). Wiley, New York.

[15] Thomas Ericsson and Axel Ruhe. 1980. The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems. *Math. Comp.* 35, 152 (1980), 1251–1268.

[16] Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. 2004. Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26, 2 (2004).

[17] Alex Gittens and Michael Mahoney. 2013. Revisiting the nystrom method for improved large-scale machine learning. In *International Conference on Machine Learning*. PMLR, 567–575.

[18] Stephan Günnemann, Ines Färber, Sebastian Raubach, and Thomas Seidl. 2013. Spectral subspace clustering for graphs with feature vectors. In *2013 IEEE 13th International Conference on Data Mining*. IEEE, 231–240.

[19] Ville Hautamaki, Ismo Karkkainen, and Pasi Franti. 2004. Outlier detection using k-nearest neighbour graph. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, Vol. 3. IEEE, 430–433.

[20] Bruce Hendrickson and Robert W. Leland. 1995. A Multi-Level Algorithm For Partitioning Graphs. *Supercomputing '95: Proceedings of the 1995 ACM/IEEE Conference on Supercomputing (CDROM)* (1995), 1–14.

[21] Sibylle Hess, Wouter Duivesteijn, Philipp Honysz, and Katharina Morik. 2019. The SpectACl of nonconvex clustering: a spectral approach to density-based clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3788–3795.

[22] Desmond J Higham and Milla Kibble. 2004. A unified view of spectral clustering. *University of Strathclyde mathematics research report* 2 (2004).

[23] Hao Huang, Shinjae Yoo, Hong Qin, and Dantong Yu. 2011. A robust clustering algorithm based on aggregated heat kernel mapping. In *2011 IEEE 11th International Conference on Data Mining*. IEEE, 270–279.

[24] Ling Huang, Donghui Yan, Nina Taft, and Michael Jordan. 2008. Spectral clustering with perturbed data. *Advances in Neural Information Processing Systems* 21 (2008).

[25] Tülin Inkaya. 2016. A Parameter-Free Similarity Graph for Spectral Clustering. *Expert Syst. Appl.* 42, 24 (dec 2016), 9489–9498. https://doi.org/10.1016/j.eswa.2015.07.074

[26] Hongjie Jia, Shifei Ding, Hong Zhu, Fulin Wu, and Lina Bao. 2013. A Feature Weighted Spectral Clustering Algorithm Based on Knowledge Entropy. *J. Softw.* 8, 5 (2013), 1101–1108.

[27] George Karypis and Vipin Kumar. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing* 20, 1 (1998), 359–392.

[28] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. 2009. Sampling techniques for the nystrom method. In *Artificial Intelligence and Statistics*. 304–311.

[29] Cornelius Lanczos. 1950. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office Los Angeles, CA.

[30] Daniel Lehmann, Liadan Ita Oćallaghan, and Yoav Shoham. 2002. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM (JACM)* 49, 5 (2002), 577–602.

[31] Richard B Lehoucq, Danny C Sorensen, and Chao Yang. 1998. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM.

[32] Mu Li, James Tin-Yau Kwok, and Baoliang Lu. 2010. Making large-scale Nyström approximation possible. In *ICML 2010-Proceedings, 27th International Conference on Machine Learning*. 631.

[33] Mu Li, Xiao-Chen Lian, James T Kwok, and Bao-Liang Lu. 2011. Time and space efficient spectral clustering via column sampling. In *CVPR 2011*. IEEE, 2297–2304.

[34] Xi Li, Weiming Hu, Chunhua Shen, Anthony Dick, and Zhongfei Zhang. 2014. Context-Aware Hypergraph Construction for Robust Spectral Clustering. *IEEE Transactions on Knowledge and Data Engineering* 26, 10 (2014), 2588–2597. https://doi.org/10.1109/TKDE.2013.126

[35] Zhenguo Li, Jianzhuang Liu, Shifeng Chen, and Xiaoou Tang. 2007. Noise robust spectral clustering. In *2007 IEEE 11th International Conference on Computer Vision*. IEEE, 1–8.

[36] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE transactions on information theory* 28, 2 (1982), 129–137.

[37] James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Vol. 1. Oakland, CA, USA, 281–297.

[38] Mahesh Mohan and Claire Monteleoni. 2017. Exploiting sparsity to improve the accuracy of Nyström-based large-scale spectral clustering. In *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 9–16.

[39] Prakash Nadkarni. 2016. Chapter 10 - Core Technologies: Data Mining and "Big Data". In *Clinical Research Computing*, Prakash Nadkarni (Ed.). Academic Press, 187–204. https://doi.org/10.1016/B978-0-12-803130-8.00010-5

[40] Andrew Ng, Michael Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems* 14 (2001), 849–856.

[41] Beresford N. Parlett. 1998. *The Symmetric Eigenvalue Problem*. Society for Industrial and Applied Mathematics, Philadelphia.

[42] Jella Pfeiffer and Franz Rothlauf. 2007. Analysis of greedy heuristics and weight-coded eas for multidimensional knapsack problems and multi-unit combinatorial auctions. In *Proceedings of the 9th annual Conference on Genetic and Evolutionary Computation*. 1529–1529.

[43] Farhad Pourkamali-Anaraki. 2020. Scalable Spectral Clustering With Nyström Approximation: Practical and Theoretical Aspects. *IEEE Open Journal of Signal Processing* 1 (2020), 242–256.

[44] Simone Romano, Nguyen Xuan Vinh, James Bailey, and Karin Verspoor. 2016. Adjusting for chance clustering comparison measures. *The Journal of Machine Learning Research* 17, 1 (2016), 4635–4666.

[45] Tomoya Sakai and Atsushi Imiya. 2009. Fast Spectral Clustering with Random Projection and Sampling. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*. Springer, 372–384.

[46] Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence* 22, 8 (2000), 888–905.

[47] G. W. Stewart. 2002. A Krylov-Schur Algorithm for Large Eigenproblems. *SIAM J. Matrix Anal. Appl.* 23, 3 (2002), 601–614. https://doi.org/10.1137/S0895479800371529

[48] Alexander Strehl and Joydeep Ghosh. 2002. Cluster Ensembles – A Knowledge Reuse Framework for Combining Multiple Partitions. *Journal on Machine Learning Research (JMLR)* 3 (December 2002), 583–617.

[49] Zhiqiang Tao, Hongfu Liu, Sheng Li, and Yun Fu. 2016. Robust spectral ensemble clustering. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 367–376.

[50] Joshua Tobin and Mimi Zhang. 2021. DCF: An Efficient and Robust Density-Based Clustering Method. In *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 629–638.

[51] Nicolas Tremblay and Andreas Loukas. 2020. Approximating spectral clustering via sampling: a review. *Sampling Techniques for Supervised or Unsupervised Tasks* (2020), 129–183.

[52] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* 17, 4 (2007), 395–416.

[53] Liang Wang, Christopher Leckie, Kotagiri Ramamohanarao, and James Bezdek. 2009. Approximate Spectral Clustering. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (Advances in Knowledge Discovery and Data Mining)*. Springer Berlin Heidelberg, 134–146.

[54] Christopher Williams and Matthias Seeger. 2001. Using the Nyström method to speed up kernel machines. In *Advances in neural information processing systems*. 682–688.

[55] He Xu, Lin Zhang, Peng Li, and Feng Zhu. 2022. Outlier detection algorithm based on k-nearest neighbors-local outlier factor. *Journal of Algorithms & Computational Technology* 16 (2022), 17483026221078111. https://doi.org/10.1177/17483026221078111

[56] Kai Zhang, Liang Lan, Jun Liu, Andreas Rauber, and Fabian Moerchen. 2012. Inductive kernel low-rank decomposition with priors: A generalized nystrom method. *arXiv preprint arXiv:1206.4619* (2012).

[57] Xiatian Zhu, Chen Change Loy, and Shaogang Gong. 2014. Constructing robust affinity graphs for spectral clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1450–1457.