Non-exhaustive, Overlapping Clustering

Joyce Jiyoung Whang*, *Member, IEEE,*
Yangyang Hou*, David F. Gleich, and Inderjit S. Dhillon, *Fellow, IEEE*

Abstract—Traditional clustering algorithms, such as K-Means, output a clustering that is disjoint and exhaustive, i.e., every single data point is assigned to exactly one cluster. However, in many real-world datasets, clusters can overlap and there are often outliers that do not belong to any cluster. While this is a well-recognized problem, most existing algorithms address either overlap or outlier detection and do not tackle the problem in a unified way. In this paper, we propose an intuitive objective function, which we call the NEO-K-Means (Non-Exhaustive, Overlapping K-Means) objective, that captures the issues of overlap and non-exhaustiveness in a unified manner. Our objective function can be viewed as a reformulation of the traditional K-Means objective, with easy-to-understand parameters that capture the degrees of overlap and non-exhaustiveness. By considering an extension to weighted kernel K-Means, we show that we can also apply our NEO-K-Means idea to overlapping community detection, which is an important task in network analysis. To optimize the NEO-K-Means objective, we develop not only fast iterative algorithms but also more sophisticated algorithms using low-rank semidefinite programming techniques. Our experimental results show that the new objective and algorithms are effective in finding ground-truth clusterings that have varied overlap and non-exhaustiveness; for the case of graphs, we show that our method outperforms state-of-the-art overlapping community detection algorithms.

Index Terms—Overlapping clustering, K-Means, outlier, semidefinite programming, graph clustering, community detection.

1 Introduction

Clustering is one of the most fundamental and important tasks in pattern analysis and recognition. Given a set of data points, the goal of clustering is to group the data points into a user-provided number of clusters such that nearby data points are assigned to the same cluster. In the traditional clustering setting, it is assumed that the clusters are pairwise disjoint and all the data points are assigned to some cluster. That is, every data point is assigned to exactly one cluster. When separations between groups are clear and the data do not contain any outliers, classical clustering methods such as K-Means [1] often succeed in appropriately grouping the data points in many realistic data models [2].

However, if the data points do not have an obvious separation and contain both outliers and large regions of overlap between groups, the traditional disjoint and exhaustive clustering methods might fail to correctly capture the underlying patterns of the data. For example, in biological gene expression data, each cluster corresponds to a group of genes which are likely to belong to the same functional class. Since genes can serve multiple functions, the underlying clusters are naturally overlapped with each other [3]. Another example is clustering a social network where the clusters (also called communities) can overlap due to the multiple personas that individuals adopt [4]. In this paper, we revisit the clustering problem from the perspective of real-world data where groups still exist but may lack clean separations and the data contain outliers. In this setting, a more reasonable goal is a non-exhaustive, overlapping clustering where a data point may be outside of any cluster, and clusters are allowed to overlap with each other.

There is substantial prior research that has examined both of these problems individually – as would be expected for an area as well studied as clustering. For example, non-exhaustive clustering is highly related to outlier detection in a dataset, which itself has an extensive literature [5]. Regarding overlap, both soft clustering [6], which only makes probabilistic assignments, and overlapping clustering models have been studied [7]. Furthermore, many variations of the K-Means algorithm have been proposed over the years [8] including recent work that considers overlapping clustering [9], [10]. We discuss the related work in more detail in Section 7 in the context of the limitations of existing methods and our new contributions. A key difference between our approach and existing ideas is that we treat the issues of non-exhaustiveness and overlap in a unified framework.

The result of our investigations is a novel improvement to the K-Means clustering objective, which we call NEO-K-Means (Non-Exhaustive, Overlapping K-Means) objective, that allows us to simultaneously identify overlapping clusters as well as outliers. The NEO-K-Means objective provides an intuitive way to handle the degree of overlap and non-exhaustiveness (the number of outliers not assigned to any cluster) while seamlessly generalizing the traditional K-Means objective. Furthermore, by considering an extension to weighted kernel K-Means, we show that our NEO-K-Means idea can also be applied to graph clustering problems. We extend a traditional normalized cut-based graph clustering objective to the non-exhaustive, overlapping clustering setting, and show that this extended

- *J.J. Whang is with the Department of Computer Science and Engineering, Sungkyunkwan University (SKKU), Suwon, South Korea. E-mail: jjwhang@skku.edu*
- *Y. Hou and D.F. Gleich are with the Department of Computer Science, Purdue University, West Lafayette, IN 47907-2107. E-mail: hou13@purdue.edu, dgleich@purdue.edu*
- *I.S. Dhilon is with the Department of Computer Science, The University of Texas at Austin, Austin, TX 78712-1757. E-mail: inderjit@cs.utexas.edu*

**Authors with equal contribution. J.J. Whang is the corresponding author.*
Manuscript received 2 Dec. 2016; revised 29 Apr. 2018.

graph clustering objective is mathematically equivalent to the weighted kernel NEO-K-Means objective with a specific weight and kernel. This equivalence enables us to apply our NEO-K-Means idea to the overlapping community detection problem in network analysis.

To optimize the NEO-K-Means objective function, we develop not only fast iterative algorithms but also more sophisticated algorithms using a low-rank semidefinite programming technique. We first present a simple iterative algorithm, which we call the NEO-K-Means algorithm, that monotonically decreases the NEO-K-Means objective function and generalizes Lloyd's method for K-Means. To provide a good initialization with the iterative NEO-K-Means algorithm, we study a convex semidefinite program (SDP) of the NEO-K-Means objective. Also, we propose a low-rank factorization of the SDP solution matrix and implement a solution procedure using an augmented Lagrangian method, which enables us to handle problems with tens of thousands of data points, providing an order of magnitude increase in scalability over the convex solver. Furthermore, we also propose two fast multiplier methods to speed the computation of the augmented Lagrangian method. Experimental results show that our new objective and algorithms are effective in finding ground-truth clusters in many real-world datasets; for the case of graphs, we show that our algorithms outperform state-of-the-art overlapping community detection methods.

This paper reflects a summary synthesis of our three conference papers on this topic [11], [12], [13]. The goal is to provide a holistic view of the ideas and to improve the depth of experimental support for our methodology. The rest of the paper is organized as follows. We introduce our NEO-K-Means objective function in Section 2, and present a connection between the NEO-K-Means objective and graph clustering problems in Section 3. We propose the simple iterative NEO-K-Means algorithm in Section 4, and present the convex SDP relaxation of the NEO-K-Means problem in Section 5. In Section 6, we present the low-rank SDP methods for NEO-K-Means. We summarize related work in Section 7, and show experimental results in Section 8. We present our conclusion in Section 9.

2 The NEO-K-Means Objective Function

Given a set of data points $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, the classic K-Means method seeks a partition of the data points into k clusters $\mathcal{C}_1, \ldots, \mathcal{C}_k$ such that they cover all the data points (formally, $\mathcal{C}_1 \cup \mathcal{C}_2 \cup \cdots \cup \mathcal{C}_k = \mathcal{X}$), and the partitions are disjoint ($\mathcal{C}_i \cap \mathcal{C}_j = \emptyset \ \forall i \neq j$). The goal of K-Means is to pick the clusters that minimize the distance from the cluster centroid, or the mean of cluster, to each of its assigned data points. The K-Means objective may be written as:

$$\min_{\{\mathcal{C}_j\}_{j=1}^k} \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in \mathcal{C}_j} \|\mathbf{x}_i - \mathbf{m}_j\|^2, \text{ where } \mathbf{m}_j = \frac{\sum_{\mathbf{x}_i \in \mathcal{C}_j} \mathbf{x}_i}{|\mathcal{C}_j|}. \quad (1)$$

It has been shown that minimizing the above objective function is an NP-hard problem even for just two clusters. However, there is an efficient heuristic K-Means algorithm [1], also known as Lloyd's algorithm, that proceeds by repeatedly assigning data points to their closest clusters and recomputing cluster centroids. This algorithm monotonically decreases the objective function.

The goal of non-exhaustive, overlapping clustering is to compute a set of cohesive clusters $\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_k$ such that $\mathcal{C}_1 \cup \mathcal{C}_2 \cup ... \cup \mathcal{C}_k \subseteq \mathcal{X}$ and the clusters need not be disjoint. To encode this new problem, we introduce an assignment matrix $\boldsymbol{U} = [u_{ij}]_{n \times k}$ where $u_{ij} \in \mathbb{B} = \{0, 1\}$ such that $u_{ij} = 1$ if $\mathbf{x}_i$ belongs to cluster j; $u_{ij} = 0$ otherwise. Using this notation, if we seek a traditional disjoint and exhaustive clustering, the number of ones in the assignment matrix $\boldsymbol{U}$ should be always equal to n because each data point is assigned to exactly one cluster. On the other hand, in a non-exhaustive, overlapping clustering, there are no restrictions on the assignment matrix $\boldsymbol{U}$; there can be multiple ones in a row, meaning that a data point can belong to multiple clusters. Also, there can be rows of all zeros, meaning that some data points can have no membership in any cluster. To control how many additional assignments we will make in $\boldsymbol{U}$, we add a constraint that the number of total assignments in $\boldsymbol{U}$ should be equal to $n + \alpha n$ where α controls the amount of overlap among the clusters. Formally, we add $\text{trace}(\boldsymbol{U}^T\boldsymbol{U}) = (1 + \alpha)n$. We require $0 \leq \alpha \leq (k - 1)$ and note that $\alpha \ll (k - 1)$ to avoid assigning each data point to every cluster.

When we only constrain the number of assignments, we observe that some data points that are relatively far from the cluster centers are not assigned to any cluster. We illustrate this behavior using a synthetic dataset shown in Figure 1(a). When we optimize an objective which only constrains the number of total assignments in $\boldsymbol{U}$, we increase overlap around the global mean as shown in Figure 1(b). To account for this bias, we also directly constrain the number of outliers which results in the NEO-K-Means objective that we define in the next paragraph (2). Optimizing (2) leads to correctly finding outliers as well as natural overlapping clustering structure as shown in Figure 1(c).

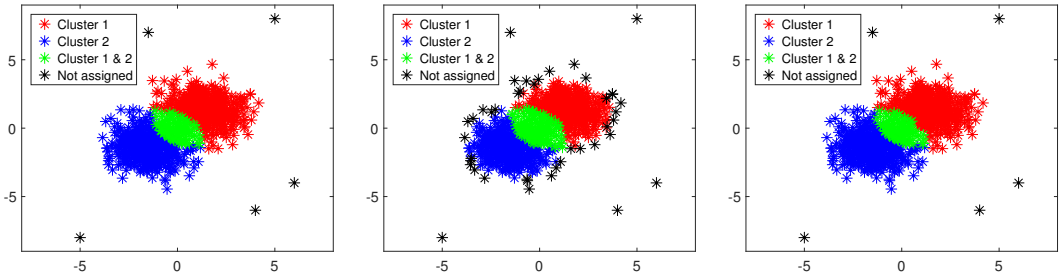
To describe how we handle outliers, we need two pieces of notation. Define an indicator function $\mathbb{I}\{exp\}$ to be $\mathbb{I}\{exp\} = 1$ if exp is true; 0 otherwise, and let $\mathbf{e}$ denote a $k \times 1$ column vector having all the elements equal to one. Then, the vector $\boldsymbol{U}\mathbf{e}$ denotes the number of clusters to which each data point belongs (note that $\boldsymbol{U}$ is the assignment matrix we defined above). Thus, $(\boldsymbol{U}\mathbf{e})_i = 0$ means that $\mathbf{x}_i$ does not belong to any cluster. By setting the upper bound of $\sum_{i=1}^n \mathbb{I}\{(\boldsymbol{U}\mathbf{e})_i = 0\}$ to be βn, we can control the non-exhaustiveness and at most βn data points can be considered as outliers. We require $0 \leq \beta n$ and note that $\beta n \ll n$ would cause most data points to be assigned to clusters. Specifically, by the definition of "outliers," βn should be a very small number compared to n.

Putting all these together, we define our NEO-K-Means objective function as follows:

$$\min_{\boldsymbol{U} \in \mathbb{B}^{n \times k}} \sum_{j=1}^{k} \sum_{i=1}^{n} u_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|^2, \text{ where } \mathbf{m}_j = \frac{\sum_{i=1}^n u_{ij}\mathbf{x}_i}{\sum_{i=1}^n u_{ij}} \quad (2)$$
$$\text{s.t. } \text{trace}(\boldsymbol{U}^T\boldsymbol{U}) = (1 + \alpha)n, \ \sum_{i=1}^n \mathbb{I}\{(\boldsymbol{U}\mathbf{e})_i = 0\} \leq \beta n.$$

Similar to K-Means, the above objective function is designed to minimize the sum of squared distances between every data point to its cluster centroid, but now the assignment is not necessarily restricted to be disjoint and exhaustive. The

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2018.2863278, IEEE Transactions on Pattern Analysis and Machine Intelligence



(a) Ground-truth clusters **(b)** NEO-K-Means without the non-exhaustiveness constraint **(c)** NEO-K-Means

Fig. 1: (a) Two ground-truth clusters are generated. Green points indicate overlap between the clusters, and black points indicate outliers. (b) When we only constrain the number of assignments, the solution contains too many false positive outliers. (c) The NEO-K-Means objective defined in (2) adds an explicit term for non-exhaustiveness that enables it to correctly cluster the data.

parameters α and β offer an intuitive way to capture the degree of overlap and non-exhaustiveness; by "turning the knob" on these parameters, the user can explore the landscape of overlapping, non-exhaustive clusterings. If α=0 and β=0, the NEO-K-Means objective function is equivalent to the standard K-means objective presented in (1). To see this, note that setting the parameter β=0 requires every data point to belong to at least one cluster, while setting α=0 makes n assignments. Thus, the resulting clustering will be disjoint and exhaustive and exactly equivalent to K-means.

We can extend (2) to a weighted kernel NEO-K-Means by introducing a nonlinear mapping ϕ and a nonnegative weight w_i for each data point $\mathbf{x}_i$ as follows:

$$\min_{\boldsymbol{U}\in\mathbb{B}^{n\times k}} \sum_{j=1}^{k}\sum_{i=1}^{n} u_{ij}w_i\|\phi(\mathbf{x}_i)-\mathbf{m}_j\|^2, \text{ where } \mathbf{m}_j=\frac{\sum_{i=1}^{n}u_{ij}w_i\phi(\mathbf{x}_i)}{\sum_{i=1}^{n}u_{ij}w_i}$$
$$\text{s.t. } \operatorname{trace}(\boldsymbol{U}^T\boldsymbol{U})=(1+\alpha)n,\ \textstyle\sum_{i=1}^{n}\mathbb{I}\{(\boldsymbol{U}\mathbf{e})_i=0\}\le\beta n. \tag{3}$$

In (3), the nonlinear mapping enables us to cluster the data points in a higher dimensional feature space ($\boldsymbol{U}$ is an assignment matrix). We can avoid forming the feature space explicitly by using the well-known kernel trick. Let $\boldsymbol{K}$ denote a kernel matrix such that $\boldsymbol{K}_{ij}=\phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j)$. The weight for each data point differentiates each data point's contribution to the objective function. Let $\boldsymbol{W}=[w_{ii}]_{n\times n}$ denote a diagonal weight matrix whose diagonal entries are equal to vertex weights, and let $\mathbf{u}_j$ denote the jth column of the assignment matrix $\boldsymbol{U}$. Then, we can rewrite (3) as follows:

$$\min_{\boldsymbol{U}\in\mathbb{B}^{n\times k}} \sum_{j=1}^{k}\left(\sum_{i=1}^{n}u_{ij}w_i\boldsymbol{K}_{ii}-\frac{\mathbf{u}_j{}^T\boldsymbol{W}\boldsymbol{K}\boldsymbol{W}\mathbf{u}_j}{\mathbf{u}_j{}^T\boldsymbol{W}\mathbf{u}_j}\right)$$
$$\text{s.t. } \operatorname{trace}(\boldsymbol{U}^T\boldsymbol{U})=(1+\alpha)n,\ \textstyle\sum_{i=1}^{n}\mathbb{I}\{(\boldsymbol{U}\mathbf{e})_i=0\}\le\beta n. \tag{4}$$

3 Graph Clustering using NEO-K-Means

It has been shown that a general weighted kernel K-Means objective is equivalent to a traditional graph clustering objective [14]. However, how to extend this idea to a *non-exhaustive, overlapping graph clustering* has remained as an open problem until we examined it in [11]. In this section, we present how we can extend the traditional graph clustering objectives to a non-exhaustive, overlapping graph clustering. This discussion provides a justification for the resulting overlapping normalized cut measure that goes beyond [11]. We show that the extended graph clustering objective is mathematically equivalent to the weighted kernel NEO-K-Means objective, which allows us to exploit the NEO-K-Means idea for solving the overlapping community detection problem.

3.1 Graph Clustering via Normalized Cut

Given a graph $G=(\mathcal{V},\mathcal{E})$, the corresponding adjacency matrix is defined as $\boldsymbol{A}=[a_{ij}]$ such that a_{ij} is equal to the edge weight between vertex i and j if there is an edge, and zero otherwise. We assume that we are working with undirected graphs where the matrix $\boldsymbol{A}$ is symmetric. We also assume that there are no self-loops in the graph, i.e., the diagonal elements of $\boldsymbol{A}$ are all zeros. The traditional graph clustering (also known as graph partitioning) problem seeks k pairwise disjoint clusters such that $\mathcal{C}_1\cup\mathcal{C}_2\cup\cdots\cup\mathcal{C}_k=\mathcal{V}$.

Normalized cut [15] is one of the most popular graph clustering objectives. Let $\texttt{links}(\mathcal{C}_p,\mathcal{C}_q)$ denote the sum of edge weights between two sets $\mathcal{C}_p$ and $\mathcal{C}_q$. Then, the normalized cut of a graph is defined as follows:

$$\texttt{NCut}(G)=\min_{\mathcal{C}_1,\mathcal{C}_2,\cdots\mathcal{C}_k}\sum_{j=1}^{k}\frac{\texttt{links}(\mathcal{C}_j,\mathcal{V}\backslash\mathcal{C}_j)}{\texttt{links}(\mathcal{C}_j,\mathcal{V})}. \tag{5}$$

Using a linear algebraic formulation, the normalized cut objective may be expressed as follows:

$$\texttt{NCut}(G)=\min_{\mathbf{y}_1,\cdots\mathbf{y}_k}\sum_{j=1}^{k}\frac{\mathbf{y}_j^T(\boldsymbol{D}-\boldsymbol{A})\mathbf{y}_j}{\mathbf{y}_j^T\boldsymbol{D}\mathbf{y}_j}=\max_{\mathbf{y}_1,\cdots\mathbf{y}_k}\sum_{j=1}^{k}\frac{\mathbf{y}_j^T\boldsymbol{A}\mathbf{y}_j}{\mathbf{y}_j^T\boldsymbol{D}\mathbf{y}_j}, \tag{6}$$

where $\boldsymbol{D}$ is the diagonal matrix of vertex degrees, and $\mathbf{y}_j$ denotes an indicator vector for cluster j, i.e., $\mathbf{y}_j(i)=1$ if a vertex i belongs to cluster j, zero otherwise; and there is no overlap or outliers.

3.2 Extending Graph Cut Objectives to Non-exhaustive, Overlapping Graph Clustering

We show how we can extend the traditional graph clustering objective to a non-exhaustive, overlapping graph clustering by exploring the implications of the normalized cut. Let us take a closer look at the normalized cut. In (5), the numerator $\texttt{links}(\mathcal{C}_j,\mathcal{V}\backslash\mathcal{C}_j)$ can be expressed as $\sum_{u\in\mathcal{C}_j}\sum_{v\in\mathcal{V}\backslash\mathcal{C}_j}\texttt{link}(u,v)$ where $\texttt{link}(u,v)$ indicates the edge weight between vertex u and v. Let $\mathcal{C}(u)$ denote a set of clusters a vertex u belongs to. Then, in disjoint and exhaustive clustering, $|\mathcal{C}(u)|=1$, and given an *undirected* edge $\{u,v\}$, the total penalty for this edge in (5) is equal to $2\,\texttt{link}(u,v)$ if $\mathcal{C}(u)\neq\mathcal{C}(v)$ since the edge is considered in $\mathcal{C}(u)$ and $\mathcal{C}(v)$, or the penalty is zero if $\mathcal{C}(u)=\mathcal{C}(v)$.

		Relationship	Total Penalty of an Edge $\{u, v\}$
$\lvert\mathcal{C}(u)\rvert = 1$	$\lvert\mathcal{C}(v)\rvert = 1$	$\mathcal{C}(u) = \mathcal{C}(v)$	0
$\lvert\mathcal{C}(u)\rvert = 1$	$\lvert\mathcal{C}(v)\rvert = 1$	$\mathcal{C}(u) \neq \mathcal{C}(v)$	$2\,\texttt{link}(u, v)$
$\lvert\mathcal{C}(u)\rvert = 1$	$\lvert\mathcal{C}(v)\rvert > 1$	$\mathcal{C}(u) \subset \mathcal{C}(v)$	$\lvert\mathcal{C}(v)\backslash\mathcal{C}(u)\rvert\,\texttt{link}(u, v)$
$\lvert\mathcal{C}(u)\rvert = 1$	$\lvert\mathcal{C}(v)\rvert > 1$	$\mathcal{C}(u) \not\subset \mathcal{C}(v)$	$(\lvert\mathcal{C}(u)\rvert + \lvert\mathcal{C}(v)\rvert)\,\texttt{link}(u, v)$
$\lvert\mathcal{C}(u)\rvert \geq 1$	$\lvert\mathcal{C}(v)\rvert = 0$	$\mathcal{C}(u) \neq \emptyset, \mathcal{C}(v) = \emptyset$	$\lvert\mathcal{C}(u)\rvert\,\texttt{link}(u, v)$
$\lvert\mathcal{C}(u)\rvert > 1$	$\lvert\mathcal{C}(v)\rvert > 1$	$\mathcal{C}(u) = \mathcal{C}(v)$	0
$\lvert\mathcal{C}(u)\rvert > 1$	$\lvert\mathcal{C}(v)\rvert > 1$	$\mathcal{C}(u) \neq \mathcal{C}(v), \mathcal{C}(u) \subset \mathcal{C}(v)$	$\lvert\mathcal{C}(v)\backslash\mathcal{C}(u)\rvert\,\texttt{link}(u, v)$
$\lvert\mathcal{C}(u)\rvert > 1$	$\lvert\mathcal{C}(v)\rvert > 1$	$\mathcal{C}(u) \bigcap \mathcal{C}(v) = \emptyset$	$(\lvert\mathcal{C}(u)\rvert + \lvert\mathcal{C}(v)\rvert)\,\texttt{link}(u, v)$
$\lvert\mathcal{C}(u)\rvert > 1$	$\lvert\mathcal{C}(v)\rvert > 1$	$\mathcal{C}(u) \not\subset \mathcal{C}(v), \mathcal{C}(u) \bigcap \mathcal{C}(v) \neq \emptyset$	$(\lvert\mathcal{C}(v)\backslash\mathcal{C}(u)\rvert + \lvert\mathcal{C}(u)\backslash\mathcal{C}(v)\rvert)\,\texttt{link}(u, v)$

TABLE 1: Possible relationships between $\mathcal{C}(u)$ and $\mathcal{C}(v)$ in a non-exhaustive, overlapping graph clustering, and the total penalty of an edge $\{u, v\}$ according to the definition of the normalized cut.

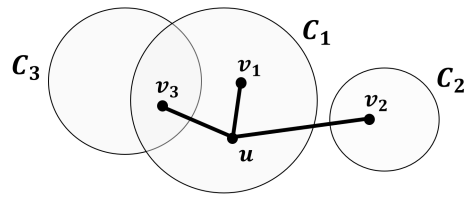


Fig. 2: The penalty of the edge $\{u, v_1\}$ is zero, the penalty of the edge $\{u, v_2\}$ is $2\,\texttt{link}(u, v_2)$, and the penalty of the edge $\{u, v_3\}$ is $\texttt{link}(u, v_3)$.

The generalization of the normalized cut objective we consider is:

$$\texttt{NCut}(G) = \min_{\mathcal{C}_1,\mathcal{C}_2,\cdots\mathcal{C}_k} \sum_{j=1}^{k} \frac{\texttt{links}(\mathcal{C}_j, \mathcal{V}\backslash\mathcal{C}_j)}{\texttt{links}(\mathcal{C}_j, \mathcal{V})}, \tag{7}$$

where now the clusters $\mathcal{C}_1, \ldots, \mathcal{C}_k$ may be non-exhaustive and overlap. In the traditional normalized cut definition, the relationship between $\mathcal{C}(u)$ and $\mathcal{C}(v)$ can be either $\mathcal{C}(u) \neq \mathcal{C}(v)$ or $\mathcal{C}(u) = \mathcal{C}(v)$. However, in a non-exhaustive, overlapping graph clustering, there can be various relationships between $\mathcal{C}(u)$ and $\mathcal{C}(v)$ because each vertex can belong to multiple clusters and there can be different patterns among the clusters the two vertices u and v belong to.

Table 1 shows the possible relationships between $\mathcal{C}(u)$ and $\mathcal{C}(v)$, and the total penalty of an edge $\{u, v\}$ according to the definition of the normalized cut (7). In general, the total penalty of an edge $\{u, v\}$ should be $(|\mathcal{C}(v)\backslash\mathcal{C}(u)| + |\mathcal{C}(u)\backslash\mathcal{C}(v)|)\,\texttt{link}(u, v)$ (this corresponds to the last row of Table 1). Indeed, the last column of each row in Table 1 can be interpreted as a special case of this general form of the penalty. Figure 2 shows some examples of the first three cases described in Table 1. In Figure 2, the penalty of the edge $\{u, v_1\}$ is zero since $\mathcal{C}(u) = \mathcal{C}(v)$ whereas the penalty of the edge $\{u, v_2\}$ is $2\,\texttt{link}(u, v_2)$ since $\mathcal{C}(u) \neq \mathcal{C}(v_2)$. On the other hand, the penalty of the edge $\{u, v_3\}$ is $\texttt{link}(u, v_3)$ because the edge $\{u, v_3\}$ is considered to be a within-cluster edge in terms of $\mathcal{C}_1$ but a between-cluster edge in terms of $\mathcal{C}_3$. This is because the vertex v_3 is in the overlapped region between $\mathcal{C}_1$ and $\mathcal{C}_3$. We see that the penalty of the edge $\{u, v_3\}$ is less than $\{u, v_2\}$, and it aligns with our intuition in that the identity of v_3 is ambiguous since it belongs to both of the clusters $\mathcal{C}_1$ and $\mathcal{C}_3$ whereas v_2 only belongs to $\mathcal{C}_2$. Since the vertex u belongs to $\mathcal{C}_1$, it seems reasonable that we impose a less penalty on the edge $\{u, v_3\}$ than $\{u, v_2\}$. For brevity, we just describe the three representative cases in detail, but one can easily extend our analysis to other cases mentioned in Table 1.

Now, we show how we can formally represent the normalized cut objective for a non-exhaustive, overlapping graph clustering. We first introduce an assignment matrix $\boldsymbol{Y} = [y_{ij}]_{n\times k}$ such that y_{ij}=1 if a vertex v_i belongs to cluster j; y_{ij}=0 otherwise. Let $\mathbf{y}_j$ denote the jth column of $\boldsymbol{Y}$. Then, we can extend (5) to a non-exhaustive, overlapping graph clustering by introducing the same constraints as in (2):

$$\begin{aligned} &\min_{\mathbf{y}_1,\cdots\mathbf{y}_k} \sum_{j=1}^{k} \frac{\mathbf{y}_j^T(\boldsymbol{D}-\boldsymbol{A})\mathbf{y}_j}{\mathbf{y}_j^T\boldsymbol{D}\mathbf{y}_j} = \max_{\mathbf{y}_1,\cdots\mathbf{y}_k} \sum_{j=1}^{k} \frac{\mathbf{y}_j^T\boldsymbol{A}\mathbf{y}_j}{\mathbf{y}_j^T\boldsymbol{D}\mathbf{y}_j} \\ &\text{s.t.}\,\text{trace}(\boldsymbol{Y}^T\boldsymbol{Y}) = (1+\alpha)n,\ \sum\nolimits_{i=1}^{n} \mathbb{I}\{(\boldsymbol{Y}\mathbf{e})_i = 0\} \leq \beta n. \end{aligned} \tag{8}$$

By adjusting α and β, we can control the degree of overlap and non-exhaustiveness. If α=0, and β=0, the above objective enforces disjoint and exhaustive clustering, thus is equivalent to the traditional normalized cut objective. Since we normalize the cut of each cluster by its volume, the upper bound of the normalized cut of a graph is k. We note that in a non-exhaustive, overlapping graph clustering, the normalized cut more gradually changes compared to the disjoint and exhaustive graph clustering because it is possible that we partially penalize edges if their endpoints belong to overlapped regions.

Even though we have focused on the normalized cut in this paper, other graph clustering objectives (e.g., ratio association [15]) also can be extended to a non-exhaustive, overlapping graph clustering using a similar approach.

3.3 Connection to the NEO-K-Means Objective

We can show that the extended normalized cut defined in (8) has a close connection to the weighted kernel NEO-K-Means objective defined in (4). In fact, by defining appropriate kernel and weight matrices in (4), we can show that (8) is equivalent to (4). Let us define the kernel as $\boldsymbol{K} \equiv \gamma\boldsymbol{W}^{-1} + \boldsymbol{W}^{-1}\boldsymbol{A}\boldsymbol{W}^{-1}$ where γ is a positive constant typically chosen to make $\boldsymbol{K}$ positive definite. Then (4) can be expressed

$$\begin{aligned} &\min_{\boldsymbol{U}} \sum_{j=1}^{k} \left(\sum_{i=1}^{n} u_{ij} w_i \frac{\gamma}{w_i} - \frac{\mathbf{u}_j{}^T\boldsymbol{A}\mathbf{u}_j}{\mathbf{u}_j{}^T\boldsymbol{W}\mathbf{u}_j} \right) \\ &= \min_{\boldsymbol{U}} \left(\gamma(1+\alpha)n - \sum_{j=1}^{k} \frac{\mathbf{u}_j{}^T\boldsymbol{A}\mathbf{u}_j}{\mathbf{u}_j{}^T\boldsymbol{W}\mathbf{u}_j} \right) = \max_{\boldsymbol{U}} \sum_{j=1}^{k} \frac{\mathbf{u}_j{}^T\boldsymbol{A}\mathbf{u}_j}{\mathbf{u}_j{}^T\boldsymbol{W}\mathbf{u}_j}. \end{aligned} \tag{9}$$

Now, in (9), let us use $\boldsymbol{W} = \boldsymbol{D}$ and notice that $\boldsymbol{U} = \boldsymbol{Y}$ in (8). Putting these together, we can see that the weighted kernel NEO-K-Means objective (4) is mathematically equivalent to the extended normalized cut objective (8). This implies that we can solve the non-exhaustive, overlapping clustering problem for both vector datasets and graph datasets in a unified framework using the NEO-K-Means idea.

4 The NEO-K-Means Algorithm

We propose the NEO-K-Means algorithm which is a simple iterative algorithm to optimize the NEO-K-Means objective function. We also propose an efficient multilevel NEO-K-Means algorithm for graph clustering.

4.1 The Iterative NEO-K-Means Algorithm

We propose a simple iterative algorithm which we call the NEO-K-Means algorithm to optimize our NEO-K-Means objective [11]. The NEO-K-Means algorithm monotonically decreases the NEO-K-Means objective until it converges to a local minimum. Having the hard constraints in (2), we make $n + \alpha n$ assignments such that at most βn data points can have no membership in any cluster. Note that the second constraint can be interpreted as follows: among n data points, at least $n - \beta n$ data points should have membership

Algorithm 1 The NEO-K-Means Algorithm

Input: $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_n\}$, the number of clusters k, the maximum number of iterations t_{max}, α, β
Output: $\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_k$

Initialize cluster means $\{\mathbf{m}_j\}_{j=1}^k$, $t = 0$.
while not converged and $t < t_{max}$ **do**
  Compute cluster means, and then compute distances between every data point and clusters $[d_{ij}]_{n\times k}$.
  Initialize $\mathcal{T} = \emptyset$, $\mathcal{S} = \emptyset$, $p = 0$, and $\bar{\mathcal{C}}_j = \emptyset$, $\hat{\mathcal{C}}_j = \emptyset\ \forall j$.
  while $p < (n + \alpha n)$ **do**
    if $p < (n - \beta n)$ **then**
      Assign $\mathbf{x}_{i^*}$ to $\bar{\mathcal{C}}_{j^*}$ such that $(i^*, j^*) = \underset{i,j}{\operatorname{argmin}}\ d_{ij}$ where $\{(i, j)\} \notin \mathcal{T}, i \notin \mathcal{S}$.
      $\mathcal{S} = \mathcal{S} \cup \{i^*\}$.
    else
      Assign $\mathbf{x}_{i^*}$ to $\hat{\mathcal{C}}_{j^*}$ such that $(i^*, j^*) = \underset{i,j}{\operatorname{argmin}}\ d_{ij}$ where $\{(i, j)\} \notin \mathcal{T}$.
    end if
    $\mathcal{T} = \mathcal{T} \cup \{(i^*, j^*)\}$.
    $p = p + 1$.
  end while
  $\forall j$, update clusters $\mathcal{C}_j = \bar{\mathcal{C}}_j \cup \hat{\mathcal{C}}_j$.
  $t = t + 1$.
end while

to some cluster. When our algorithm makes assignments of points to clusters, it uses two phases to satisfy these two constraints. Thus, each cluster $\mathcal{C}_j$ decomposes into two sets $\bar{\mathcal{C}}_j$ and $\hat{\mathcal{C}}_j$ that record the assignments made in each phase.

Algorithm 1 describes the NEO-K-Means algorithm. We first initialize cluster centroids (We will discuss how we initialize the clusters in Section 5 and Section 6). Given cluster centroids, we compute all the distances $[d_{ij}]_{n\times k}$ between every data point and clusters, and for every data point, record its closest cluster and that distance. Then, the data points are sorted in ascending order by the distance to its closest cluster. To ensure at least $n - \beta n$ data points are assigned to some cluster (i.e., to satisfy the second constraint), we assign the first $n-\beta n$ data points to their closest clusters. Let $\bar{\mathcal{C}}_j$ denote the assignments made by this step. Thus, $\sum_{j=1}^k |\bar{\mathcal{C}}_j| = n - \beta n$. Then, we make $\beta n + \alpha n$ more assignments by taking $\beta n + \alpha n$ minimum distances among $[d_{ij}]_{n\times k}$ such that $\mathbf{x}_i \notin \bar{\mathcal{C}}_j$. Let $\hat{\mathcal{C}}_j$ denote the assignments made by this step. Thus, $\sum_{j=1}^k |\hat{\mathcal{C}}_j| = \beta n + \alpha n$. Finally, $\sum_{j=1}^k (|\bar{\mathcal{C}}_j| + |\hat{\mathcal{C}}_j|) = n + \alpha n$. Once all the assignments are made, we update cluster centroids by recomputing the mean of each cluster. We repeat this procedure until the change in the objective function is sufficiently small or the maximum number of iterations is reached.

Note that the algorithm does not forcibly choose βn points as outliers; indeed, the number of outliers is *less* than or equal to βn, and depends on the the distances between data points and their "secondary" clusters. We note that, if $\alpha = 0$ and $\beta = 0$, then the NEO-K-Means algorithm is identical to the standard K-Means algorithm [1]. In this scenario, all the n data points are assigned to their closest clusters. We can show that the NEO-K-Means algorithm guarantees monotonic decrease in the objective function as the following result shows:

Theorem 1 ([11]). *Algorithm 1 monotonically reduces the NEO-K-Means objective in (2) while satisfying the constraints specified by α and β.*

Proof. Let $J^{(t)}$ denote the objective at the t-th iteration. Then,

$$
\begin{aligned}
J^{(t)} &= \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in \mathcal{C}_j^{(t)}} \|\mathbf{x}_i - \mathbf{m}_j^{(t)}\|^2 \\
&\geq \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in \bar{\mathcal{C}}_j^{(t+1)}} \|\mathbf{x}_i - \mathbf{m}_j^{(t)}\|^2 + \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in \hat{\mathcal{C}}_j^{(t+1)}} \|\mathbf{x}_i - \mathbf{m}_j^{(t)}\|^2 \\
&= \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in \mathcal{C}_j^{(t+1)}} \|\mathbf{x}_i - \mathbf{m}_j^{(t)}\|^2 \text{ since } \mathcal{C}_j^{(t+1)} = \bar{\mathcal{C}}_j^{(t+1)} \cup \hat{\mathcal{C}}_j^{(t+1)} \\
&\geq \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in \mathcal{C}_j^{(t+1)}} \|\mathbf{x}_i - \mathbf{m}_j^{(t+1)}\|^2 \text{ (property of centroids)} \\
&= J^{(t+1)}
\end{aligned}
$$

The first inequality follows from the update scheme used to form $\bar{\mathcal{C}}_j$ and $\hat{\mathcal{C}}_j$ by our algorithm, and the second inequality follows from the property of the cluster centroids. Clearly the algorithm always maintains feasibility, i.e., the constraints specified by the parameters α and β are always satisfied. □

4.2 The Multilevel NEO-K-Means Algorithm for Graphs

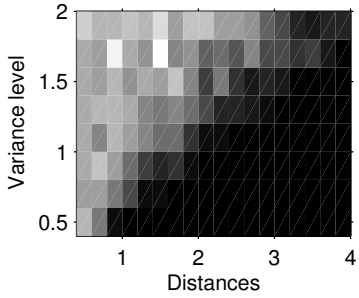
In Section 3, we show that the normalized cut can be extended to a non-exhaustive, overlapping graph clustering, and the *extended normalized cut objective* is equivalent to the *weighted kernel NEO-K-Means objective*. This implies that we can solve the non-exhaustive, overlapping graph clustering problem using the weighted kernel NEO-K-Means algorithm. The difference between the standard NEO-K-Means algorithm and the weighted kernel NEO-K-Means algorithm is how to compute the distance between a data point and a cluster. Using our definitions of kernel and weights (presented in Section 3.3), the distance between a vertex v_i and a cluster $\mathcal{C}_j$ can be quantified as follows:

$$
\begin{aligned}
&\texttt{dist}(v_i, \mathcal{C}_j) = \\
&-\frac{2\,\texttt{links}(v_i, \mathcal{C}_j)}{\deg(v_i)\deg(\mathcal{C}_j)} + \frac{\texttt{links}(\mathcal{C}_j, \mathcal{C}_j)}{\deg(\mathcal{C}_j)^2} + \frac{\gamma}{\deg(v_i)} - \frac{\gamma}{\deg(\mathcal{C}_j)}
\end{aligned} \tag{10}
$$

where $\deg(v_i)$ denotes the degree of vertex v_i, and $\deg(\mathcal{C}_j)$ denotes the sum of edge weights of vertices in $\mathcal{C}_j$. Then, Algorithm 1 can be applied to graph data by computing the distances using (10).

When we solve the graph clustering problem, we can exploit a multilevel framework similar to [14] to achieve a better objective function value. In the multilevel framework, an input graph is coarsened by merging adjacent vertices level by level. As a result, a series of smaller graphs are created. Once the input graph becomes small enough to be directly partitioned, an initial partitioning is performed. The clustering result of the coarsest level graph is projected onto the graph at the level above it. Many different heuristics can be used at any of these coarsening or projection stages in order to improve the overall performance. The clustering is then refined through a refinement algorithm which plays the most important role in optimizing an objective function. Once the clustering is refined, the clustering result is projected onto the graph at the level above it, and then refined. This procedure is repeated until we reach the original graph.

While we use similar heuristics for the coarsening and the initial partitioning phases as in [14], we implement



For this two-class clustering experiment where class means are separated at a given distance and each class has a given variance, the rank of $\boldsymbol{Z}$ is represented in gray scale where black is rank 2 (the ideal value). If the clusters are separated well without too much overlap, then we recover the precise rank. Away from this case, the rank increases showing the effect of the relaxation.

Fig. 3: The rank of the co-occurrence matrix $\boldsymbol{Z}$.

the weighted kernel NEO-K-Means algorithm for the refinement phase. In this way, we optimize the weighted kernel NEO-K-Means objective function at multiple scales. We observe that this multilevel NEO-K-Means algorithm is able to efficiently optimize the extended normalized cut objective function.

5 An SDP For NEO-K-Means

The iterative NEO-K-Means algorithm is *fast*, but suffers from the classic problem that iterative algorithms for K-Means fall into local minimizers given poor initialization of the clusters. Thus, it is important to provide a good initialization with the NEO-K-Means algorithm. To find a good initialization, we study a convex semidefinite program (SDP) relaxation of the NEO-K-Means objective.

Semidefinite programs are one of the most general classes of tractable convex optimization problems. Once we formulate a convex relaxation of the NEO-K-Means objective, the convex problem can be globally optimized in time and memory that is polynomial in the input size. The relaxed solution can then be rounded to a discrete assignment solution which can provide a good initialization for the iterative NEO-K-Means algorithm.

5.1 Formulation

We begin by stating an exact SDP-like program for the weighted kernel NEO-K-Means objective defined in (4) and then describe how to relax it to an SDP. We use the same notation as the previous section. The essential idea with the SDP-like version is that we replace the assignment matrix $\boldsymbol{U}$ with a normalized cluster co-occurrence matrix $\boldsymbol{Z}$:

$$\boldsymbol{Z} = \sum_{j=1}^{k} \frac{\boldsymbol{W}\mathbf{u}_j(\boldsymbol{W}\mathbf{u}_j)^T}{\mathbf{u}_j^T\boldsymbol{W}\mathbf{u}_j}. \tag{11}$$

When $\boldsymbol{Z}$ is defined from an assignment matrix $\boldsymbol{U}$, then values of Z_{ij} are non-zero when items co-occur in a cluster. With appropriate constraints on the matrix $\boldsymbol{Z}$, it serves as a direct replacement for the assignment matrix $\boldsymbol{U}$.

To state the problem, let $\boldsymbol{K}$ denote the kernel matrix of the data points, e.g., if $\boldsymbol{X}$ is the data matrix whose rows correspond to data vectors, then $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}^T$ is just the simple linear kernel matrix. Let $\mathbf{d}$ be a vector where $d_i = w_iK_{ii}$, i.e., a weighted diagonal from $\boldsymbol{K}$. We need two new types of variables as well:

- Let $\mathbf{f}$ denote a vector of length n such that the ith entry indicates the number of clusters data point i belongs to.
- Let $\mathbf{g}$ denote a vector of length n such that the ith entry is one if the data point i belongs to any clusters, and zero if the data point does not belong to any cluster.

The following program is equivalent to the weighted kernel NEO-K-Means objective with a discrete assignment matrix:

$$\begin{array}{lll} \underset{\boldsymbol{Z},\mathbf{f},\mathbf{g}}{\text{maximize}} & \operatorname{trace}(\boldsymbol{K}\boldsymbol{Z}) - \mathbf{f}^T\mathbf{d} & \\ \text{subject to} & \operatorname{trace}(\boldsymbol{W}^{-1}\boldsymbol{Z}) = k, & (a) \\ & Z_{ij} \geq 0, & (b) \\ & \boldsymbol{Z} \succeq 0, \boldsymbol{Z} = \boldsymbol{Z}^T & (c) \\ & \boldsymbol{Z}\mathbf{e} = \boldsymbol{W}\mathbf{f}, & (d) \\ & \mathbf{e}^T\mathbf{f} = (1+\alpha)n, & (e) \\ & \mathbf{e}^T\mathbf{g} \geq (1-\beta)n, & (f) \\ & \mathbf{f} \geq \mathbf{g}, & (g) \\ & \operatorname{rank}(\boldsymbol{Z}) = k, & (h) \\ & \mathbf{f} \in \mathcal{Z}^n_{\geq 0}, \mathbf{g} \in \{0,1\}^n. & (i) \end{array} \tag{12}$$

Constraints $(a), (b), (c)$, and (h) encode the fact that $\boldsymbol{Z}$ must arise from an assignment matrix. Constraints $(d), (e), (f), (g)$, and (i) express the amount of overlap and non-exhaustiveness in the solution. This is a mixed-integer, rank constrained SDP. As such, it is combinatorially hard to optimize just like the original NEO-K-Means objective.

The constraints that make this a combinatorial problem are (h) and (i). If we relax these constraints:

$$\begin{array}{ll} \underset{\boldsymbol{Z},\mathbf{f},\mathbf{g}}{\text{maximize}} & \operatorname{trace}(\boldsymbol{K}\boldsymbol{Z}) - \mathbf{f}^T\mathbf{d} \\ \text{subject to} & (a), (b), (c), (d), (e), (f), (g) \\ & 0 \leq \mathbf{f} \leq k, 0 \leq \mathbf{g} \leq 1 \end{array} \tag{13}$$

then we arrive at a convex problem. Thus, any local optimal solution of (13) must be a global solution. Note that the constraint (a) acts equivalently to a nuclear norm constraint on the rank since $\boldsymbol{Z}$ is a symmetric positive semi-definite matrix and $\boldsymbol{W}$ is a positive diagonal matrix. Solving (13) requires a black-box SDP solver such as CVX [16]. As we convert the NEO-K-Means objective into a standard form of SDP, the number of variables becomes $\mathcal{O}(n^2)$ and the resulting complexity is worse than $\mathcal{O}(n^3)$ in most cases, and can be as bad as $\mathcal{O}(n^6)$. In Section 5.4, we examine the scalability of several well-known SDP solvers.

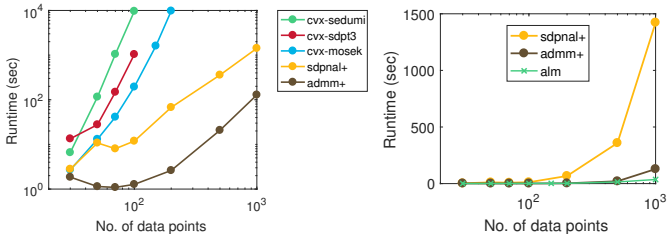
5.2 Rounding solutions

Once we get a real-valued co-occurrence matrix $\boldsymbol{Z}$, we need to convert it into a binary assignment matrix $\boldsymbol{U}$ through a rounding procedure. Since we know that the idealized rank of $\boldsymbol{Z}$ is k, we can factorize $\boldsymbol{Z}$ by $\boldsymbol{Z} = \boldsymbol{Y}\boldsymbol{Y}^T$ by applying a symmetric nonnegative matrix factorization [17] where $\boldsymbol{Y}$ is an $n \times k$ non-negative matrix. After the $\boldsymbol{Z}$ matrix is represented by a low-rank matrix $\boldsymbol{Y}$, we can convert it to a discrete solution by the procedure discussed in Section 6.3.

5.3 Rank of the Relaxed Solution Matrix

Since we formulate the co-occurrence matrix $\boldsymbol{Z}$ according to (11), we know that the rank of the matrix $\boldsymbol{Z}$ should be k. We conduct a simple experiment to investigate the empirical rank of the co-occurrence matrix from solving the SDP (13). We generate synthetic datasets using the Gaussian distribution. We assume there are two clusters with overlap. As we change the distance between the cluster centroids and the variance level of each cluster, we solve and compute the rank of the solution $\boldsymbol{Z}$. In Figure 3, we present the rank of the solution matrix in gray scale. Black indicates the rank of $\boldsymbol{Z}$ is 2, and lighter grey colors show large ranks. We see that when the two clusters are reasonably separable from each other (i.e., larger distance between the cluster centroids or

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2018.2863278, IEEE Transactions on Pattern Analysis and Machine Intelligence



(a) Run times of SDP solvers on synthetic datasets **(b)** The fastest SDP solvers and the LRSDP solver

Fig. 4: Run times of SDP solvers and the LRSDP solver on synthetic datasets with different numbers of data points.

smaller variance level), the rank of the solution matrix is close to two—the number of clusters.

5.4 Scalability of SDP solvers

We test the run time of several SDP solvers such as MOSEK, SDPT3, SEDUMI, and SDPNAL+ [18] to understand the limits of current SDP solution procedures. We can also solve (13) by an alternating direction method of multipliers (the convergent three-block ADMM procedure due to [19]), which is denoted by ADMM+ (we only run 400 ADMM iterations, which may not have converged, but gives a useful approximation). It is well known that SDP solvers are not highly scalable. Our goal is simply to understand the practical limits to this approach. We measure the running time of each solver on synthetic datasets with different numbers of data points. Figure 4(a) shows these results where SDPNAL+ and ADMM+ are the fastest methods and could handle problems with thousands of data points in less than a day. To deal with larger datasets, we can consider a low-rank factorization of the SDP solution matrix (as we briefly mentioned in the previous subsection) and implement a solution procedure using an augmented Lagrangian method. We discuss this low-rank SDP (LRSDP) method in Section 6. In Figure 4(b), we show the run times of the fastest SDP solvers, SDPNAL+ and ADMM+, and our LRSDP solver (denoted by ALM) for comparison. We see that the LRSDP solver is much faster than the SDP solvers.

6 A Low-Rank SDP For NEO-K-Means

We propose optimizing a low-rank factorization of the SDP solution matrix to tackle large-scale SDPs [20]. The resulting optimization problem is a quadratically constrained problem with a quadratic objective that can no longer be globally optimized. An augmented Lagrangian procedure, for instance, will only converge to a local minimizer. Nevertheless, when this approach has been used in the past with high-quality optimization methods, it frequently generates solutions that are *as good as the global optimal* from convex solvers [20], a fact which has some theory [21].

We discuss how to formulate the low-rank SDP (LRSDP) for the NEO-K-Means objective, and also present several algorithms to optimize this non-linear problem. Our LRSDP algorithms can handle problems with tens of thousands of data points, providing an order of magnitude increase in scalability over the convex solver. On the problems where we can compare with the convex formulations, we achieve globally optimal objective values.

6.1 Formulation

In the SDP formulation of the NEO-K-Means objective (13), the matrix $\boldsymbol{Z}$ should ideally be rank k, although this cannot be enforced. If we represent $\boldsymbol{Z}$ as $\boldsymbol{Y}\boldsymbol{Y}^T$ where $\boldsymbol{Y}$ is an $n \times k$ non-negative matrix, then we can enforce this consraint. The following optimization problem is a low-rank SDP for (13) (we have chosen to write it in the standard form of a minimization problem with explicit slack variables $\mathbf{s}$ and r to convert the inequality constraints into equality and bound constraints).

$$\begin{array}{lll} \underset{\boldsymbol{Y},\mathbf{f},\mathbf{g},\mathbf{s},r}{\text{minimize}} & \mathbf{f}^T\mathbf{d} - \text{trace}(\boldsymbol{Y}^T\boldsymbol{K}\boldsymbol{Y}) & \\ \text{subject to} & k = \text{trace}(\boldsymbol{Y}^T\boldsymbol{W}^{-1}\boldsymbol{Y}) & (s) \\ & 0 = \boldsymbol{Y}\boldsymbol{Y}^T\mathbf{e} - \boldsymbol{W}\mathbf{f} & (t) \\ & 0 = \mathbf{e}^T\mathbf{f} - (1+\alpha)n & (u) \\ & 0 = \mathbf{f} - \mathbf{g} - \mathbf{s} & (v) \\ & 0 = \mathbf{e}^T\mathbf{g} - (1-\beta)n - r & (w) \\ & Y_{ij} \geq 0, \mathbf{s} \geq 0, r \geq 0 & \\ & 0 \leq \mathbf{f} \leq k\mathbf{e}, 0 \leq \mathbf{g} \leq 1 & \end{array} \tag{14}$$

We replace the constraint $\boldsymbol{Y}\boldsymbol{Y}^T \geq 0$ with the stronger constraint $\boldsymbol{Y} \geq 0$. This problem is a quadratic programming problem with quadratic constraints. Even though we lose convexity by formulating the low rank SDP, this nonlinear programming problem only requires $\mathcal{O}(nk)$ memory and existing nonlinear programming techniques allow us to scale to large problems.

6.2 Solvers for the NEO-K-Means LRSDP

We present three solution procedures for the NEO-K-Means LRSDP problem: a classical augmented Lagrangian method, a proximal augmented Lagrangian method, and an alternating direction method of multipliers.

6.2.1 *Classical Augmented Lagrangian Method (ALM)*

To solve (14), we use an augmented Lagrangian framework. This is an iterative strategy where each step consists of minimizing an augmented Lagrangian of the problem that includes a current estimate of the Lagrange multipliers for the constraints as well as a penalty term that drives the solution towards the feasible set. Augmented Lagrangian techniques have been successful in previous studies of low-rank SDP approximations [20].

Let $\lambda = [\lambda_1; \lambda_2; \lambda_3]$ be the Lagrange multipliers associated with the three scalar constraints $(s), (u), (w)$, and $\boldsymbol{\mu}$ and $\boldsymbol{\gamma}$ be the Lagrange multipliers associated with the vector constraints (t) and (v), respectively. Let $\sigma \geq 0$ be a penalty parameter. The augmented Lagrangian for (14) is

$$\begin{aligned} &\mathcal{L}_A(Y, \mathbf{f}, \mathbf{g}, \mathbf{s}, r; \lambda, \boldsymbol{\mu}, \boldsymbol{\gamma}, \sigma) = \\ &\quad \underbrace{\mathbf{f}^T\mathbf{d} - \text{trace}(\boldsymbol{Y}^T\boldsymbol{K}\boldsymbol{Y})}_{\text{the objective}} \\ &\quad - \lambda_1(\text{trace}(\boldsymbol{Y}^T\boldsymbol{W}^{-1}\boldsymbol{Y}) - k) + \frac{\sigma}{2}(\text{trace}(\boldsymbol{Y}^T\boldsymbol{W}^{-1}\boldsymbol{Y}) - k)^2 \\ &\quad - \boldsymbol{\mu}^T(\boldsymbol{Y}\boldsymbol{Y}^T\mathbf{e} - \boldsymbol{W}\mathbf{f}) + \frac{\sigma}{2}(\boldsymbol{Y}\boldsymbol{Y}^T\mathbf{e} - \boldsymbol{W}\mathbf{f})^T(\boldsymbol{Y}\boldsymbol{Y}^T\mathbf{e} - \boldsymbol{W}\mathbf{f}) \\ &\quad - \lambda_2(\mathbf{e}^T\mathbf{f} - (1+\alpha)n) + \frac{\sigma}{2}(\mathbf{e}^T\mathbf{f} - (1+\alpha)n)^2 \\ &\quad - \boldsymbol{\gamma}^T(\mathbf{f} - \mathbf{g} - \mathbf{s}) + \frac{\sigma}{2}(\mathbf{f} - \mathbf{g} - \mathbf{s})^T(\mathbf{f} - \mathbf{g} - \mathbf{s}) \\ &\quad - \lambda_3(\mathbf{e}^T\mathbf{g} - (1-\beta)n - r) + \frac{\sigma}{2}(\mathbf{e}^T\mathbf{g} - (1-\beta)n - r)^2. \end{aligned} \tag{15}$$

At each step in the augmented Lagrangian solution framework, we solve the following subproblem

$$\begin{array}{ll} \text{minimize} & \mathcal{L}_A(\boldsymbol{Y}, \mathbf{f}, \mathbf{g}, \mathbf{s}, r; \lambda, \boldsymbol{\mu}, \boldsymbol{\gamma}, \sigma) \\ \text{subject to} & Y_{ij} \geq 0, \mathbf{s} \geq 0, r \geq 0, \\ & 0 \leq \mathbf{f} \leq k\mathbf{e}, 0 \leq \mathbf{g} \leq 1. \end{array} \tag{16}$$

We use limited-memory BFGS with bound constraints [22] to minimize the subproblem with respect to the variables $\boldsymbol{Y}$, $\mathbf{f}$, $\mathbf{g}$, $\mathbf{s}$ and r. This requires the gradient of $\mathcal{L}_A$ with respect to the variables, see [12] for the derivation.

6.2.2 Proximal Augmented Lagrangian Method (PALM)

The proximal augmented Lagrangian method differs from the classical augmented Lagrangian method only in an additional proximal regularization term for primal updates. This can be considered as a type of simultaneous primal-dual proximal-point step that helps to regularize the subproblems solved at each step. This leads to the iteration

$$\mathbf{x}^{t+1} = \operatorname*{argmin}_{\mathbf{x}} \mathcal{L}_{\mathcal{A}}(\mathbf{x}; \boldsymbol{\lambda}^t, \boldsymbol{\mu}^t, \boldsymbol{\gamma}^t, \sigma) + \frac{1}{2\tau}\|\mathbf{x} - \mathbf{x}^t\|^2,$$

where $\mathbf{x}$ represents $[\mathbf{y}; \mathbf{f}; \mathbf{g}; \mathbf{s}; r]$ with $\mathbf{y} = \boldsymbol{Y}(:)$ vectorized by column. Then we update the multipliers $\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\gamma}$ as in the classical augmented Lagrangian. We may also need to update the penalty parameter σ and the proximal parameter τ respectively.

We use a limited-memory BFGS with bound constraints to solve the new subproblem with respect to the variable $\mathbf{x}$. If we let $\tau = \sigma$, this special case is called the proximal method of multipliers, first introduced in [23]. The proximal method of multipliers has better theoretical convergence guarantees for convex optimization problems compared with the classical augmented Lagrangian [23]. In the non-convex setting like our NEO-K-Means problem, we believe it is likely to help to improve conditioning of the Hessian in the subproblems and thus reduce the solution time for each subproblem. And this is indeed what we find.

Since we use the proximal augmented Lagrangian on the problem without any convexity, local convergence is the best we can achieve. By specializing a general convergence result about the proximal augmented Lagrangian in [24] to our problem (14), we can show the local convergence of our algorithm. The resulting theorem is a general convergence result about the proximal augmented Lagrangian method for non-convex problem with bound constrained subproblems. Further details are available in [13].

6.2.3 Alternating Direction Method of Multipliers (ADMM)

There are four sets of variables in problem (14) ($\boldsymbol{Y}$, $\mathbf{f}$, $\mathbf{g}$ and slack variables). We can use this structure to break the augmented Lagrangian function into smaller subproblems for each set of variables. Some of these subproblems are then easier to solve. For example, updating variable $\mathbf{f}$ alone is a simple convex problem, thus it is very efficient to have a globally optimal solution. The alternating direction method of multipliers approach of updating block variables $\boldsymbol{Y}$, $\mathbf{f}$, $\mathbf{g}$, $\mathbf{s}$ and r respectively, utilizes this property, which leads to the following iterates:

$$\begin{aligned}
\boldsymbol{Y}^{t+1} &= \operatorname*{argmin}_{\boldsymbol{Y}} \mathcal{L}_{\mathcal{A}}(\boldsymbol{Y}, \mathbf{f}^t, \mathbf{g}^t, \mathbf{s}^t, r^t; \boldsymbol{\lambda}^t, \boldsymbol{\mu}^t, \boldsymbol{\gamma}^t, \sigma) \\
\mathbf{f}^{t+1} &= \operatorname*{argmin}_{\mathbf{f}} \mathcal{L}_{\mathcal{A}}(\boldsymbol{Y}^{t+1}, \mathbf{f}, \mathbf{g}^t, \mathbf{s}^t, r^t; \boldsymbol{\lambda}^t, \boldsymbol{\mu}^t, \boldsymbol{\gamma}^t, \sigma) \\
\mathbf{g}^{t+1} &= \operatorname*{argmin}_{\mathbf{g}} \mathcal{L}_{\mathcal{A}}(\boldsymbol{Y}^{t+1}, \mathbf{f}^{t+1}, \mathbf{g}, \mathbf{s}^t, r^t; \boldsymbol{\lambda}^t, \boldsymbol{\mu}^t, \boldsymbol{\gamma}^t, \sigma) \\
\mathbf{s}^{t+1} &= \operatorname*{argmin}_{\mathbf{s}} \mathcal{L}_{\mathcal{A}}(\boldsymbol{Y}^{t+1}, \mathbf{f}^{t+1}, \mathbf{g}^{t+1}, \mathbf{s}, r^t; \boldsymbol{\lambda}^t, \boldsymbol{\mu}^t, \boldsymbol{\gamma}^t, \sigma) \\
r^{t+1} &= \operatorname*{argmin}_{r} \mathcal{L}_{\mathcal{A}}(\boldsymbol{Y}^{t+1}, \mathbf{f}^{t+1}, \mathbf{g}^{t+1}, \mathbf{s}^{t+1}, r; \boldsymbol{\lambda}^t, \boldsymbol{\mu}^t, \boldsymbol{\gamma}^t, \sigma)
\end{aligned}$$

Algorithm 2 Rounding $\boldsymbol{Y}$ to a binary matrix $\boldsymbol{U}$

Input: $\boldsymbol{Y}$, $\boldsymbol{W}$, $\mathbf{f}$, $\mathbf{g}$, α, β
Output: $\boldsymbol{U}$

```
Update Y = W^{-1} Y.
Set D to be the largest (n − βn) coordinates of g.
for each entry i in D do
    Set S to be the top ⌊f_i⌋ entries in Y(i, :).
    Set U(i, S) = 1.
end for
Set f̄ = f − ⌊f⌋.
Set R to be the largest entries in f̄.
for each entry i in R do
    Pick a cluster ℓ where Y(i, ℓ) is the maximun over all clusters
    where i is not currently assigned.
    Set U(i, ℓ) = 1.
end for
```

then the multipliers $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, $\boldsymbol{\gamma}$ and the penalty parameter σ are updated accordingly.

We expect that this strategy will aid convergence because it decouples the update of $\boldsymbol{Y}$ from the update of $\mathbf{f}$. In the problem with all variables, the interaction of these terms has the strongest non-convex interaction. We now detail how we solve each of the subproblems.

Update $\boldsymbol{Y}$. We use a limited-memory BFGS with bound constraints to solve the subproblem with respect to the variables $\boldsymbol{Y}$ since it is non-convex.

Update $\mathbf{f}$ and $\mathbf{g}$. The update for $\mathbf{f}$ and $\mathbf{g}$ respectively both have the following general form:

$$\begin{aligned}
\underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) = \mathbf{x}^T\mathbf{a} + \frac{\sigma}{2}\mathbf{x}^T\boldsymbol{D}\mathbf{x} + \frac{\sigma}{2}(\mathbf{e}^T\mathbf{x})^2 \\
\text{subject to} \quad & 0 \le \mathbf{x} \le b\mathbf{e}
\end{aligned} \tag{17}$$

where $\boldsymbol{D}$ is a positive diagonal matrix, $\mathbf{a}$ is a constant vector, and b is a constant. To solve this, we use ideas similar to [25, S6.2.5]. Let $\tau = \mathbf{e}^T\mathbf{x}$, thus $0 \le \tau \le bn$. We solve this problem by finding roots of the following function $F(\tau)$:

$$F(\tau) = \tau - \mathbf{e}^T P[-\tfrac{1}{\sigma}\boldsymbol{D}^{-1}(\mathbf{a} + \sigma\tau\mathbf{e}); 0, b]$$

where the function $P[\mathbf{x}; 0, b]$ projects the point $\mathbf{x}$ onto the rectangular box $[0, b]$. To find these roots, bisection suffices because $F(0) \le 0$ and $F(bn) \ge 0$. This is a globally optimal solution. A detailed analysis can be found in [13].

Update $\mathbf{s}$ and r. These updates just require solving one variable quadratic optimization with simple bound constraints; the result is a simple update procedure.

6.3 Rounding Procedure

The LRSDP solvers presented in Section 6.2 produce real-valued solutions. Thus, we need to convert $\boldsymbol{Y}$ into a binary assignment matrix $\boldsymbol{U}$ through a rounding procedure. Indeed, $\boldsymbol{Y}$ can be regarded as the normalized assignment matrix as follows:

$$\boldsymbol{Y} = \boldsymbol{W}\hat{\boldsymbol{U}} \tag{18}$$

where $\hat{\boldsymbol{U}} = [\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, \ldots, \hat{\mathbf{u}}_k]$, and $\hat{\mathbf{u}}_c = \mathbf{u}_c / \sqrt{\mathbf{u}_c^T\boldsymbol{W}\mathbf{u}_c}$ for any $c = 1, \ldots, k$. Thus, one way to get $\boldsymbol{U}$ from $\boldsymbol{Y}$ is to select the top $(1+\alpha)n$ entries in $\boldsymbol{W}^{-1}\boldsymbol{Y}$ as the clustering assignment. We find this strategy useful when we solve the weighted kernel NEO-K-Means problem.

We also notice that both the vectors $\mathbf{f}$ and $\mathbf{g}$ provide important information about the solution. Namely, $\mathbf{f}$ gives us a good approximation to the number of clusters each data point is assigned to, and $\mathbf{g}$ indicates which data points are not assigned to any cluster. By utilizing this information,

TABLE 2: The objective values and the run time (in seconds) of SDP (italics) and our LRSDP (bold) solvers on real-world data.

Method	Dolphins		Les Miserables		music	
	Obj.	Time	Obj.	Time	Obj.	Time
mosek	70.4311	47.36	88.4600	89.76	—	—
sdpnal+	70.0313	13.49	88.0635	23.06	64823.5	437.44
admm+	70.0339	2.23	88.0593	2.47	65885.1	38.32
alm	70.0317	2.40	88.0649	5.02	64824.4	107.68
palm	70.0342	1.97	88.0801	2.87	64823.3	43.53
admm	70.0323	1.65	88.1052	1.43	64823.9	29.14

we propose Algorithm 2 to derive $\boldsymbol{U}$ from $\boldsymbol{Y}$. It uses the largest $n - \beta n$ entries of the vector $\mathbf{g}$ to determine the set of data points to assign first. Each data point i is assigned to $\lfloor f_i \rfloor$ clusters based on the values in the ith row of $\boldsymbol{Y}$. The remaining assignments are all based on the largest residual elements in $\mathbf{f} - \lfloor \mathbf{f} \rfloor$.

6.4 Comparison of the SDP and the Low-Rank SDP

We present the SDP and the low-rank SDP (LRSDP) formulations of our NEO-K-Means objective function, and also discuss the solution procedures in the previous sections. Now we compare the performance of the SDP solvers and our proposed LRSDP solvers on real-world datasets. We consider two graph clustering problems using 'dolphins' [26] and 'les miserables' [27] datasets. The 'dolphins' network represents frequent associations between 62 dolphins (there are 159 undirected edges in the network), and 'les miserables' network represents the co-appearance of characters in the novel Les Miserables (there are 77 nodes and 254 edges). The 'music' dataset contains 593 music songs, each of which is represented by a 72 dimensional feature vector (details are available in Section 8.1.1).

Table 2 shows the objective function values and the run times (MOSEK fails on the 'music' dataset due to the size). We report the objective values before the relaxed solution is rounded to a discrete assignment solution to precisely measure how much our LRSDP solutions are different from the solution returned by the SDP solvers. For the ADMM+ method, we limit the maximum iteration to 400 so that it serves as a fast heuristic. Thus, ADMM+ had not yet converged on the larger dataset 'music' yielding a larger objective value in Table 2. We can see that the objective values returned from the SDP solver SDPNAL+ and returned from the proposed LRSDP solvers are essentially identical (they differ up to the solution tolerance given by the methods). In these cases, then, we are successful in finding a globally optimal solution (i.e., (13) and (14) are the same in these cases). Also, when we compare the run times, we find that the LRSDP solvers are faster than the SDP solvers. Note that the two fast multiplier methods, PALM and ADMM, are faster than the classical augmented Lagrangian method (ALM).

7 Related Work

Both the aspects of overlap and non-exhaustiveness in clustering have been studied before, albeit rarely considered together in a unified manner as we do. We recognize that [28] also considers both overlap and non-exhaustiveness by modifying the traditional K-medoids algorithm, but their methodologies include complicated heuristics.

A few recent papers study the clustering problem with outlier detection. In particular, [29] have proposed the `k-means--` algorithm, which discovers clusters and outliers in a unified fashion; however it does not find overlapping clusters. We focus our discussion on overlapping clustering as that literature is the most closely related to our contribution. Soft clustering methods, such as fuzzy K-Means [6], relax the binary assignment constraint and replace it by a probabilistic assignment. Thresholding these probabilities may result in both overlapping assignments to clusters and non-exhaustive partitions, although it is difficult to control these effects.

There have been many attempts to extend K-Means to overlapping clustering. For example, [9] defines *okm* by introducing a concept of the mean of cluster centers each data point belongs to. Also, [10] has reformulated the *okm* objective function by adding a sparsity constraint, which results in proposing explicit/implicit sparsity constrained clustering. On the other hand, from the Bayesian perspective, [7] proposed a generative model, called *moc*, where each data point is assumed to be generated from an exponential family. We compare the performance of these methods with our NEO-K-Means method in Section 8.1.

In the context of graph clustering, many different types of overlapping graph clustering, or overlapping community detection methods, have been presented. We previously developed a method called *nise* using a seed expansion idea [30]. In this algorithm, good seeds are detected in a network, and the seeds are expanded using a personalized PageRank scheme. Compared to this seed-and-grow algorithm, our NEO-K-Means algorithm adopts a more principled approach. Among the existing methods, the scalable alternatives include *demon* [31] and *bigclam* methods [32]; which we compare against. We also compare with *oslom* [33] which detects outliers and produces statistically significant overlapping communities. Many other methods are discussed in a recent survey [34], although the majority of successful approaches tend to suffer scalability issues on large networks like those we consider. Our derivation of the relationship between NEO-K-Means and overlapping community detection is inspired by [14] which showed the connection between K-Means and graph partitioning.

Our SDP formulations are related to convex relaxations of the traditional K-Means objective [35], [36], [37]. For instance, [35] employs the same general strategy of using a low-rank factorization of the SDP for K-Means in concert with an augmented Lagrangian solver for the resulting nonlinear optimization problem. Even more generally, our work fits into the broad setting of convex relaxations of clustering problems including normalized cut objectives [38].

Using augmented Lagrangian methods to solve low-rank factorizations of SDP solutions has a long history of delivering successful performance when the data arise from graphs. For instance, [20] originally proposed this idea for the MAX-CUT and minimum bisection SDPs. Later, similar ideas were used to address key weaknesses in spectral clustering [39] on power-law graphs.

8 Experimental Results

We show experimental results of our NEO-K-Means method on various real-world datasets, and compare the perfor-

mance with state-of-the-art overlapping clustering and overlapping community detection methods.

8.1 Data Clustering

We first evaluate the performance of the NEO-K-Means algorithm on a data clustering task.

8.1.1 Datasets

We use six different real-world datasets from [40] and [41] to measure the performance of different clustering algorithms. Table 3 presents some basic statistics of these datasets. On these datasets, we have the ground-truth clusters and 'avg. csize' denotes the average size of the ground-truth clusters.

The first three columns of Table 3 correspond to datasets from computer vision applications [40]. In these datasets, we have a set of images where each image is represented by a high-dimensional feature vector and annotated by various attributes such as parts (e.g., "arm", "wing", etc.) and materials (e.g., "glass", "plastic", etc.). We treat each attribute as a cluster, which results in forming overlapping clusters of the images since an image can have multiple attributes. We create three different datasets by selecting different attributes such that the ground-truth clusters contain overlap. In particular, for the vision1 dataset, we select images that are labelled by "wood" or "glass" or both. For the vision2 dataset, we select images that are annotated by at least one of the following attributes: "Nose", "Hair", "Arm", "Foot/Shoe", "Metal", and "Furry". Similarly, for the vision3 dataset, we consider "Mouth", "Face", "Nose" "Hair", "Wood", "Glass", and "Horn".

For example, the vision1 dataset contains 390 images with two ground-truth clusters, each of which corresponds to "wood" and "glass". Among the 390 images, 111 images belong to both of the clusters. We run the iterative NEO-K-Means algorithm with $k = 2$ on this dataset, and select some representative images from the resulting clusters in Figure 5. Since clustering is unsupervised, we do not get a label for each cluster, but by looking at the clustering result, we realize that the images which are assigned to both of the clusters are usually annotated by both "wood" and "glass" as shown in Figure 5(a) whereas most of the images which are only assigned to one of the clusters are annotated by either "wood" or "glass" but not both as shown in Figure 5(b) & Figure 5(c). We further validate this approach in more scenarios in our more systematic studies below.

We also use some benchmark datasets of multi-label learning from [41]. The 'scene' dataset [42] is a set of scene image feature vectors where each image is labelled by their scenes, e.g., beach, sunset, mountain, etc. Each scene label can be considered as a ground-truth cluster, and the clusters can overlap because an image can contain more than one scene. The 'yeast' dataset [3] is from a biology domain. This dataset contains a set of feature vectors for genes where each feature vector is constructed based on micro-array expression data and phylogenetic profiles of genes, and each gene is labelled by its functional classes. Since a gene can belong to multiple functional classes, each gene can have multiple labels. The 'music' dataset [43] consists of a set of feature vectors extracted from 593 different music songs. In this dataset, each song is labelled by emotions presented in the song, e.g., happy, surprised, relaxing, etc. Since several different emotions can be expressed in a song, a song can have more than one label. On these datasets, we treat each label as a ground-truth cluster, which enables us to interpret these multi-labelled datasets as datasets having overlapping ground-truth clusters.

TABLE 3: Real-world vector datasets.

	vision1	vision2	vision3	scene	yeast	music
n	390	1,792	915	2,407	2,417	593
dimension	9,751	9,751	9,751	294	103	72
avg. csize	250.5	566.3	275.9	430.8	731.5	184.7
k	2	6	7	6	14	6

(a) Wood & Glass **(b)** Wood **(c)** Glass

Fig. 5: Representative images from the clusters produced by the NEO-K-Means method. The images on the overlapped region are annotated by both "wood" and "glass" whereas the images which are not on the overlapped region are annotated by either "wood" or "glass".

8.1.2 Parameter Selection

The NEO-K-Means method includes two important parameters α and β. Now, we discuss some guidelines for how to select reasonable α and β values in practice.

Indeed, parameter selection is considered to be a challenging task, so many existing clustering algorithms leave this as an open problem. For example, in `k-means--` algorithm [29], the number of outliers is a required input. Most other clustering methods (e.g., [6], [10]) also have their own model parameters that should be set by a user. While some model parameters of other clustering methods tend to be non-intuitive to set or it can be hard to predict the effect of a particular parameter setting, the NEO-K-Means parameters α and β are intuitive parameters that allow users to specify how much overlap/non-exhaustiveness they want. So, users may be able to estimate these parameters from the domain knowledge. If they are unknown, we can estimate α and β values by using the heuristics discussed below.

Choosing β. To estimate the non-exhaustiveness parameter β, we first run the traditional K-Means algorithm [1]. Let d_i denote the distance between data point i and its closest cluster. We compute the mean (denoted by μ) and the standard deviation (denoted by σ) of d_i (i=1,$\cdots$,n). If a distance d_i is greater than $\mu + 3\sigma$, then we consider the data point i as an outlier. That is, we consider a data point to be an outlier if the distance to its closest cluster is greater than three standard deviations from the mean by following the *three sigma rule* in statistics [44]. In this way, we can estimate the number of outliers, and thus get the β value.

Choosing α. Different datasets might contain different levels of overlap between the clusters. Note that $0 \leq \alpha \ll (k$-1 as we described in Section 2. Since α controls the amount of overlap, the choice of α should depend on whether a user expects a small overlap, a medium overlap, or a large overlap between the clusters. For a small number of clusters (e.g., $k \approx 10$), we suggest to use α=0.1, α=1, and α=$\sqrt{k}$-1.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2018.2863278, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 4: The NEO-K-Means objective function values and run times of the iterative NEO-K-Means algorithm with four different initializations on the 'yeast' dataset.

		best	worst	mean	std.
Objective value	kmeans+neo	9494	9610	9549	51
	lrsdp-alm+neo	9280	9414	9370	58
	lrsdp-palm+neo	9280	9414	9363	50
	lrsdp-admm+neo	9280	9398	9354	48
Run time (sec.)	kmeans+neo	31	60	40	11
	lrsdp-alm+neo	4083	4462	4301	144
	lrsdp-palm+neo	1549	1802	1633	100
	lrsdp-admm+neo	914	1202	1029	107

For a large number of clusters (e.g., $k \geq 100$), we suggest to try α=1/($\sqrt{k}$-1), α=1/($\log k$-1), α=1, α=$\log k$-1, and α=$\sqrt{k}$-1.

8.1.3 Improvement from Advanced Optimization

In Section 4, we present the simple iterative NEO-K-Means algorithm, and in Section 6, we propose using a low-rank SDP (LRSDP) technique to provide a good initialization with the iterative NEO-K-Means algorithm. We show that the low-rank SDP problem for NEO-K-Means can be solved by the classical augmented Lagrangian method (ALM in short), or via even faster methods, PALM and ADMM, in Section 6.2. Table 4 shows the NEO-K-Means objective function values and run times of the iterative NEO-K-Means algorithm with four different initializations (the traditional K-Means, ALM, PALM, and ADMM) on the 'yeast' dataset. We run each method 5 times and report the best, the worst, the mean and the standard deviation. We see that by applying the LRSDP methods, we can significantly lower the objective function values, i.e., we can produce qualitatively better clustering results by using the LRSDP initialization. When we compare the run times, we see that both PALM and ADMM methods are faster than ALM. Thus, we empirically observe that the two fast multiplier methods are useful to reduce the run time of solving the LRSDP NEO-K-Means problem with no change in quality. Note that we do not expect any of the optimization-based methods will be faster than the simple iterative K-Means method since it is a completely different type of algorithm. Even though we only report the results on the 'yeast' dataset for brevity, we were able to observe similar results on other datasets.

8.1.4 Clustering Performance via Ground-truth

We compare the NEO-K-Means method with other overlapping clustering methods – *moc* [7], *okm* [9], and explicit/implicit sparsity constrained clustering [10] (denoted by *esp* and *isp*, respectively). For *moc*, *esp*, and *isp*, we used the software and default parameters provided by the authors of [7] and [10]. For the NEO-K-Means method, we estimate the β value by the strategy described in Section 8.1.2, and set three different α values, each of which corresponds to a small, a medium, and a large overlap (we set α=0.1, α=1, and α=$\sqrt{k}$-1). These are denoted by NEOs, NEOm, and NEOl, respectively. On the vision1 dataset, we just use α=0.1 and α=$\sqrt{k}$-1 because k=2.

To evaluate the resulting set of clusters from each method, we compute the average F1 score [32], the pairwise F1 score [7], and the average NMI scores [45]. Each of these metrics is considered to be a standard metric to measure how well a clustering algorithm finds the ground-truth clusters. For the average F1 score and the pairwise F1 score, we use the exactly same definitions used in [32] and [7], respectively. The average NMI score is computed by averaging the NMI scores of the best match between the ground-truth clusters and the algorithmic clusters. Given a set of algorithmic clusters $\mathcal{C}$, and the ground-truth clusters $\mathcal{S}$, the NMI score of a single ground-truth cluster S_i is defined as $\text{NMI}(\mathcal{S}_i) = \text{NMI}(\mathcal{S}_i, \mathcal{C}_{j^*})$ such that $j^* = \text{argmax}_j \text{NMI}(\mathcal{S}_i, \mathcal{C}_j)$ where $\mathcal{C}_j \in \mathcal{C}$. Similarly, we can also define the NMI score of a single algorithmic cluster $\text{NMI}(\mathcal{C}_i)$ by picking the ground-truth cluster $\mathcal{S}_j \in \mathcal{S}$ with the highest NMI score. Then, the average NMI is computed as follows:

$$\text{Avg. NMI} = \frac{1}{2}\Big\{\frac{1}{|\mathcal{S}|}\sum_{\mathcal{S}_i \in \mathcal{S}} \text{NMI}(\mathcal{S}_i) + \frac{1}{|\mathcal{C}|}\sum_{\mathcal{C}_i \in \mathcal{C}} \text{NMI}(\mathcal{C}_i)\Big\}.$$

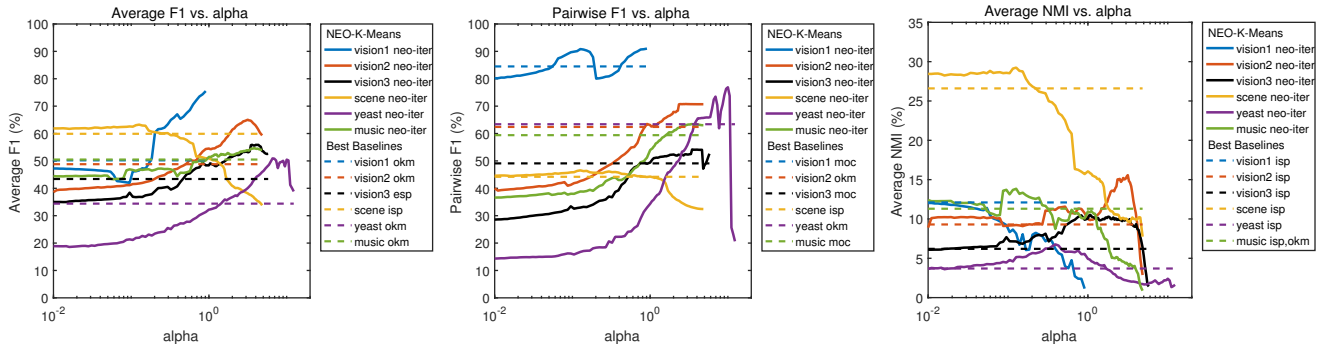
This is similar to how the average F1 is computed in [32].

TABLE 5: The average F1, pairwise F1, and NMI scores.

		Average F1 (%)			Pairwise F1 (%)			Average NMI (%)		
		best	worst	avg.	best	worst	avg.	best	worst	avg.
vision1	*moc*	39.9	39.9	39.9	84.5	84.5	84.5	8.8	8.8	8.8
	esp	50.1	48.9	49.4	83.5	82.6	83.2	11.8	11.1	11.4
	isp	47.2	47.2	47.2	79.4	79.4	79.4	**12.1**	**12.1**	**12.1**
	okm	50.2	50.2	50.2	83.7	83.7	83.7	11.5	11.5	11.5
	NEOs	42.2	42.2	42.2	**89.9**	**89.9**	**89.9**	9.3	9.3	9.3
	NEOl	**67.9**	**67.9**	**67.9**	83.5	83.5	83.5	6.6	6.6	6.6
vision2	*moc*	34.6	31.4	32.9	64.7	41.9	57.3	6.7	4.4	5.4
	esp	47.8	45.4	46.9	60.1	58.0	58.6	7.3	5.8	6.4
	isp	40.7	38.4	39.4	39.7	37.7	38.6	10.6	7.3	9.3
	okm	49.9	46.4	48.8	63.5	61.2	62.4	7.8	6.5	7.1
	NEOs	42.4	40.6	41.4	44.8	41.6	43.2	10.9	8.3	9.3
	NEOm	55.8	54.2	54.7	64.2	62.6	63.1	**11.3**	10.0	10.3
	NEOl	**56.3**	**56.2**	**56.2**	**65.0**	**64.8**	**64.9**	11.1	**11.0**	**11.1**
vision3	*moc*	32.4	30.0	31.2	53.0	43.5	49.1	6.3	4.6	5.7
	esp	43.9	42.3	43.4	47.0	45.9	46.6	4.9	4.3	4.7
	isp	37.1	34.4	35.5	30.8	28.1	29.5	7.2	5.7	6.2
	okm	43.9	42.0	42.8	47.6	45.8	46.5	6.2	5.6	6.0
	NEOs	38.4	36.0	37.3	34.2	32.6	33.3	7.7	6.1	6.9
	NEOm	49.7	49.6	49.6	52.8	52.7	52.7	**11.4**	**11.3**	**11.3**
	NEOl	**51.4**	**51.3**	**51.4**	**54.0**	**54.0**	**54.0**	10.3	10.1	10.2
scene	*moc*	44.1	41.0	42.3	37.4	36.8	37.0	20.5	19.3	19.9
	esp	55.5	52.7	54.4	41.3	40.3	40.7	20.3	17.6	19.3
	isp	61.7	**58.8**	59.9	45.7	42.5	44.2	28.2	**25.3**	26.6
	okm	57.5	55.2	56.2	43.1	42.1	42.5	23.3	20.5	21.8
	NEOs	**62.5**	58.4	**60.5**	**46.0**	**44.6**	**45.5**	**29.1**	24.9	**27.1**
	NEOm	51.5	51.3	51.4	44.7	44.5	44.6	16.4	16.2	16.3
	NEOl	48.5	47.7	48.1	43.0	42.5	42.8	14.8	13.8	14.3
yeast	*moc*	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	esp	34.4	32.8	33.6	63.7	58.7	61.4	1.6	1.4	1.5
	isp	19.9	18.6	19.0	14.2	13.8	14.0	4.0	3.6	3.7
	okm	34.8	34.1	34.4	63.9	62.5	63.4	2.5	1.8	2.0
	NEOs	21.7	20.7	21.3	16.4	15.8	16.1	4.7	4.7	4.7
	NEOm	35.7	35.6	35.6	39.0	38.9	38.9	**7.3**	**7.3**	**7.3**
	NEOl	**42.2**	**40.7**	**41.9**	**65.9**	**61.0**	**64.9**	2.6	2.6	2.6
music	*moc*	50.6	47.1	49.4	**60.6**	58.6	**59.4**	10.8	8.6	9.9
	esp	51.2	49.1	50.2	59.3	57.7	58.5	10.5	8.7	9.5
	isp	50.1	47.2	48.4	52.9	45.6	49.6	12.7	10.6	11.3
	okm	51.6	49.7	50.5	57.7	54.8	56.4	11.9	**10.8**	11.3
	NEOs	46.8	43.8	45.0	42.5	38.0	40.1	**13.4**	10.2	**11.9**
	NEOm	51.4	51.3	51.3	54.3	54.2	54.2	11.0	**10.8**	10.9
	NEOl	**52.6**	**52.5**	**52.6**	58.9	**58.7**	58.7	9.0	8.8	8.9

We run each of the algorithms 5 times and Table 5 shows the results (we present the best, the worst, and the average scores for each method). If an algorithm happens to return empty clusters or clusters that contain all the data points, we exclude these clusters when we compute the metric scores. On the 'yeast' dataset, *moc* returns 13 empty clusters, and one cluster that contains all the data points. So we cannot report the score of *moc* on this dataset. In Table 5, we can see that the NEO-K-Means method shows the highest scores in most of the cases. Indeed, among 54 cases (6 datasets, 3 metrics, and 3 scores), the NEO-K-Means achieves the highest score on 48 cases. We note that the NMI metric gives

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2018.2863278, IEEE Transactions on Pattern Analysis and Machine Intelligence



(a) Average F1 vs. α **(b)** Pairwise F1 vs. α **(c)** Average NMI vs. α

Fig. 6: The clustering performance of the iterative NEO-K-Means method with different α values.

a slightly different ranking of the algorithms from the F1 metrics. For example, on the vision2 dataset, *isp* is worse than *okm* in terms of the pairwise F1 score, but better than *okm* in terms of the NMI score. On the other hand, the NEO-K-Means method consistently outperforms other methods in terms of all the three metrics. We conclude that the NEO-K-Means objective and the algorithm are beneficial to detect the underlying patterns of real-world data, and thus useful to identify ground-truth overlapping clusters.

Finally, we conduct a sensitivity analysis for the parameter α of the NEO-K-Means algorithm (note that the β value can be automatically estimated by the strategy described in Section 8.1.2). Figure 6 shows how the clustering performance of the iterative NEO-K-Means algorithm changes as we change the α values. We try 100 different α values ranging from 0.01 to (k-1). For each α, we run the algorithm five times and pick the result which is associated with the smallest NEO-K-Means objective function. For this experiment, we initialize the NEO-K-Means algorithm with the fast traditional K-Means method instead of the LRSDP method because we need to run the algorithm for 500 times on each dataset. Therefore, in Figure 6, we expect that each of the lines can be slightly boosted up if we use the LRSDP initialization. When we evaluate the clustering, we filter out an empty cluster as well as a cluster that contains all the data points. A different color indicates a different dataset, and the dotted line indicates the performance of the best baseline method on that dataset. By comparing the performance of the NEO-K-Means with different α values and that of the best baseline method in terms of the three metrics on the six datasets, we can see a reasonable range of α values that allows the NEO-K-Means method to outperform the best baseline method.

8.2 Graph Clustering

Now, we show experimental results of our weighted kernel NEO-K-Means algorithm on graph data.

8.2.1 Case Study with the Karate Club Network

As an illustration of our method, we first apply our multilevel NEO-K-Means algorithm (discussed in Section 4.2) on Zachary's Karate Club network which represents friendship relationships among 34 members where node 1 and node 34 are known to be the instructor and the student founder of the club, respectively. These two nodes are central in the network forming two natural clusters around them. We run the NEO-K-Means with α=0.2, and β=0, so in this setting, the algorithm will make 41 assignments in total, i.e., 7 nodes can belong to both clusters (note that the number of common friends of node 1 and node 34 is four, so we are looking for something a little less than twice the obvious overlap). Figure 7 shows the clustering result of the multilevel NEO-K-Means. We see that the nodes that are assigned to both clusters have strong interactions with both of the underlying clusters meaning that the NEO-K-Means is able to reveal the natural underlying overlapping structure of the network.

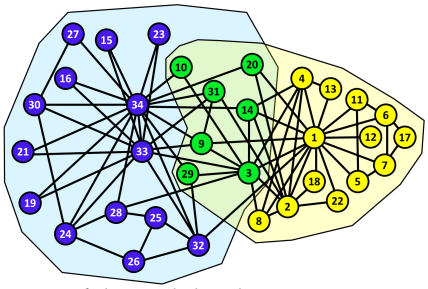
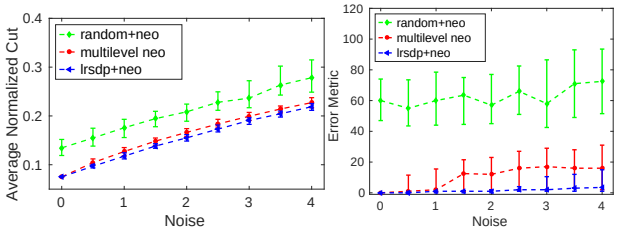


Fig. 7: Output of the multilevel NEO-K-Means algorithm.



(a) Average Normalized Cut **(b)** Error Metric

Fig. 8: A synthetic study of overlapping community detection on a Watts-Strogatz cycle graph: (a) & (b) show the normalized cut and the error measure returned by the iterative NEO-K-Means algorithm with random initialization, the multilevel NEO-K-Means algorithm, and the iterative NEO-K-Means algorithm with the LRSDP initialization.

8.2.2 Multilevel NEO-K-Means and the iterative NEO-K-Means with LRSDP Initialization on a Small Synthetic Graph

In Section 6, we discuss that we can provide a good initialization with the iterative NEO-K-Means algorithm using the LRSDP method. Also, in Section 4.2, we develop a multilevel NEO-K-Means algorithm by exploiting the problem structure. We expect that the LRSDP method produces more accurate and reliable clusterings than the multilevel algorithm in the regime of medium-scale problems (e.g., tens of thousands of data points). This regime is ideal because the optimization based method is more computationally expensive than the multilevel algorithm, which is an efficient procedure designed for problems with millions of

data points. To see the difference between these methods, we study the behavior on a synthetic problem with community detection on a cycle graph. The graph is a Watts-Strogatz random graph with 100 nodes where each node has edges to five neighbors on each side of the cycle. We also add random edges based on an Erdös-Rényi graph with an expected degree, which we consider as noise edges. When the noise is low, clusterings should respect the cycle structure and be continuous, connected regions. Hence, we compute an error measure for each cluster based on the number of points disconnected from the largest connected component in the cycle. On this dataset, we compare three methods: the iterative NEO-K-Means algorithm with random initialization (denoted by *random+neo*), the multilevel NEO-K-Means algorithm (denoted by *multilevel neo*), and the iterative NEO-K-Means algorithm with the LRSDP initialization using the ALM method (denoted by *lrsdp+neo*). We run 100 trials and plot the the median, 25th and 75th percentiles of the normalized cut scores and the number of disconnected nodes by varying the noise level. Figure 8(a) & Figure 8(b) show the results. By comparing with the performance of *random+neo*, we see the effectiveness of *multilevel neo* and *lrsdp+neo*. Note that *lrsdp+neo* achieves the best performance in terms of both the normalized cut and the number of disconnected nodes.

8.2.3 Community Detection on Real-world Networks

We compare the performance of the NEO-K-Means method with other state-of-the-art overlapping community detection methods: *demon* [31], *oslom* [33], *bigclam* [32], and *nise* [30]. We use seven different real-world networks from [46] as shown in Table 6. For *nise*, we use the "Graclus centers" seeding strategy and the Fiedler PageRank clustering scheme. For the NEO-K-Means, we set $\beta = 0$ on these datasets because we expect that one can apply graph-based pre-processing techniques that remove obvious outliers (e.g., connected components analysis). We determine α values such that the output contains a similar amount of community overlap with the *nise* method. We use the multilevel NEO-K-Means algorithm (denoted by NEOml) and the LRSDP-based NEO-K-Means (denoted by NEOlr).

Also, we try a triangle-cut-based [47] multilevel NEO-K-Means algorithm (denoted by NEOmlt). In this approach, we use the weighted adjacency matrix $A_W = A^2 \circ A$ where $\circ$ denotes a Hadamard product of the matrices and A is the original adjacency. Then, $[A_W]_{ij}$ indicates the number of triangles that use the edge $\{i, j\}$. Using this weighted graph A_W as an input of the multilevel NEO-K-Means, we are able to improve the performance of the NEO-K-Means on larger datasets that are hard to be processed by the LRSDP approach. This choice corresponds to using triangles instead of edges in the definition of conductance [47]. We use the conductance (denoted by cnd), the weighted triangle conductance [47] (denoted by wcnd) and the modularity (denoted by mod) metrics to gauge the quality of the communities returned by each of the algorithms. These metrics are considered to be standard ways to measure the cohesiveness of the clusters. Since a good overlapping community detection method should cover a large portion of the graph with a set of cohesive clusters, we compute AUC (Area Under the Curve) of the metric-versus-coverage. See [30] for the details about how to compute the AUC score. In Table 7, a higher AUC score indicates a better clustering result. We notice that *demon* and *oslom* cannot process the Orkut and the LiveJournal networks because of the scalability issue. We see that the NEO-K-Means algorithm achieves the highest AUC on all the networks, which indicates that it produces the most cohesive groups of communities.

TABLE 6: Real-world graph datasets.

	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$		$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$
Facebook	348	2,866	Amazon	334,863	925,872
HepPh	11,204	117,619	Orkut	731,332	21,992,171
AstroPh	17,903	196,972	LiveJournal	1,757,326	42,183,338
CondMat	21,363	91,286			

TABLE 7: AUC scores (%) on real-world graphs in terms of conductance, weighted conductance, and modularity. A higher AUC score indicates a better clustering result.

		demon	*oslom*	*bigclam*	*nise*	NEOml	NEOmlt	NEOlr
FB	cnd	50.5	68.1	17.0	70.3	71.5	77.0	**80.9**
	wcnd	49.4	57.2	17.9	69.8	74.0	73.3	**76.1**
	mod	6.7	9.3	0.9	9.4	10.3	10.4	**11.7**
HP	cnd	49.7	53.5	37.5	89.8	88.7	89.7	**91.7**
	wcnd	46.5	44.0	17.1	90.4	92.9	93.4	**95.1**
	mod	3.9	0.7	0.9	18.9	16.9	19.4	**19.5**
AP	cnd	43.0	42.0	35.5	84.7	86.1	82.9	**86.3**
	wcnd	43.1	36.1	29.7	88.3	90.5	88.8	**90.9**
	mod	2.2	0.1	0.5	16.4	**18.3**	16.9	**18.3**
CM	cnd	43.0	57.2	51.3	89.5	91.2	90.2	**91.4**
	wcnd	44.2	55.2	49.3	90.0	92.2	**92.8**	92.3
	mod	0.1	<0.1	0.6	19.6	20.6	20.3	**20.7**
AZ	cnd	52.0	76.4	68.4	94.1	**95.0**	90.6	N/A
	wcnd	62.1	84.9	73.1	97.5	**98.0**	95.3	N/A
	mod	<0.1	<0.1	<0.1	4.9	9.1	**14.6**	N/A
OK	cnd	N/A	N/A	20.4	71.9	82.1	**90.9**	N/A
	wcnd	N/A	N/A	28.4	85.8	90.7	**95.3**	N/A
	mod	N/A	N/A	<0.1	5.0	8.1	**19.5**	N/A
LJ	cnd	N/A	N/A	4.7	**82.2**	78.3	78.4	N/A
	wcnd	N/A	N/A	6.0	86.8	87.5	**87.8**	N/A
	mod	N/A	N/A	<0.1	13.3	9.0	**15.4**	N/A

9 Conclusion

Overall, these experiments demonstrate that our non-exhaustive, overlapping clustering framework has the best performance in terms of finding the ground-truth clusters among a large class of state-of-the-art methods. We conclude that NEO-K-Means is a useful algorithm to analyze the complex data in current data-centric applications.

Acknowledgments

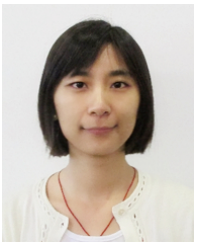
This research was supported by Basic Science Research Program through the NRF of Korea funded by MOE(2016R1D1A1B03934766 and NRF-2010-0020210) to JW, by NSF CAREER award CCF-1149756, NSF IIS-1546488, NSF Center for Science of Information STC (CCF-093937), DARPA SIMPLEX, and Sloan Research Fellowship to DG, and by NSF grants CCF-1320746 and IIS-1546452 to ID.

References

[1] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.
[2] A. Kumar and R. Kannan, "Clustering with spectral norm and the k-means algorithm," in *FOCS*, 2010, pp. 299–308.
[3] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification," in *NIPS*, 2001, pp. 681–687.
[4] R. Burt, *Structural Holes: The Social Structure of Competition*. Harvard University Press, 1995.
[5] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, 2009.

[6] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: The fuzzy c-means clustering algorithm," *Computers & Geosciences*, vol. 10, 1984.
[7] A. Banerjee, C. Krumpelman, J. Ghosh, S. Basu, and R. J. Mooney, "Model-based overlapping clustering," in *SIGKDD*, 2005.
[8] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, pp. 651–666, 2010.
[9] G. Cleuziou, "An extended version of the k-means method for overlapping clustering," in *ICPR*, 2008, pp. 1–4.
[10] H. Lu, Y. Hong, W. N. Street, F. Wang, and H. Tong, "Overlapping clustering with sparseness constraints," in *ICDM Workshops*, 2012.
[11] J. J. Whang, I. S. Dhillon, and D. F. Gleich, "Non-exhaustive, overlapping k-means," in *SDM*, 2015, pp. 936–944.
[12] Y. Hou, J. J. Whang, D. F. Gleich, and I. S. Dhillon, "Non-exhaustive, overlapping clustering via low-rank semidefinite programming," in *SIGKDD*, 2015, pp. 427–436.
[13] ——, "Fast multiplier methods to optimize non-exhaustive, overlapping clustering," in *SDM*, 2016, pp. 297–305.
[14] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *TPAMI*, vol. 29, no. 11, pp. 1944–1957, 2007.
[15] J. Shi and J. Malik, "Normalized cuts and image segmentation," *TPAMI*, vol. 22, no. 8, pp. 888–905, 2000.
[16] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," http://cvxr.com/cvx.
[17] A. Vandaele, N. Gillis, Q. Lei, K. Zhong, and I. S. Dhillon, "Coordinate descent methods for symmetric nonnegative matrix factorization," *CoRR*, 2015. [Online]. Available: http://arxiv.org/abs/1509.01404
[18] L. Yang, D. Sun, and K.-C. Toh, "Sdpnal+: a majorized semismooth newton-cg augmented lagrangian method for semidefinite programming with nonnegative constraints," *Mathematical Programming Computation*, vol. 7, no. 3, pp. 331–366, 2015.
[19] D. Sun, K.-C. Toh, and L. Yang, "A convergent 3-block semiproximal alternating direction method of multipliers for conic programming with 4-type constraints," *SIAM J. Optim.*, vol. 25, 2015.
[20] S. Burer and R. D. Monteiro, "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization," *Mathematical Programming*, vol. 95, pp. 329–357, 2003.
[21] ——, "Local minima and convergence in low-rank semidefinite programming," *Mathematical Programming*, vol. 103, 2005.
[22] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, 1995.
[23] R. T. Rockafellar, "Augmented Lagrangians and applications of the proximal point algorithm in convex programming," *Math. Oper. Res.*, vol. 1, no. 2, pp. 97–116.
[24] T. Pennanen, "Local convergence of the proximal point algorithm and multiplier methods without monotonicity," *Math. Oper. Res.*, vol. 27, no. 1, pp. 170–191, 2002.
[25] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Opt.*, vol. 1, no. 3, pp. 127–239, 2014.
[26] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, "The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations: Can geographic isolation explain this unique trait?" *Behavioral Ecology and Sociobiology*, vol. 54, pp. pp. 396–405, 2003.
[27] D. E. Knuth, *The Stanford GraphBase: A Platform for Combinatorial Computing*. Addison-Wesley, 1993.
[28] Y.-L. Chen and H.-L. Hu, "An overlapping cluster algorithm to provide non-exhaustive clustering," *EJOR*, vol. 173, 2006.
[29] S. Chawla and A. Gionis, "k-means--: A unified approach to clustering and outlier detection," in *SDM*, 2013, pp. 189–197.
[30] J. J. Whang, D. F. Gleich, and I. S. Dhillon, "Overlapping community detection using neighborhood-inflated seed expansion," *TKDE*, vol. 28, no. 5, pp. 1272–1284, 2016.
[31] M. Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi, "Demon: a local-first discovery method for overlapping communities," in *SIGKDD*, 2012, pp. 615–623.
[32] J. Yang and J. Leskovec, "Overlapping community detection at scale: a nonnegative matrix factorization approach," in *WSDM*, 2013, pp. 587–596.
[33] A. Lancichinetti, F. Radicchi, J. Ramasco, and S. Fortunato, "Finding statistically significant communities in networks," *PLOS ONE*, vol. 6, no. 4, 2011.
[34] J. Xie, S. Kelley, and B. K. Szymanski, "Overlapping community detection in networks: the state of the art and comparative study," *ACM Computing Surveys*, vol. 45, no. 4, pp. 43:1–43:35, 2013.
[35] B. Kulis, A. C. Surendran, and J. C. Platt, "Fast low-rank semidefinite programming for embedding and clustering," in *International Conference on Artifical Intelligence and Statistics*, 2007, pp. 235–242.
[36] J. Peng, "0-1 semidefinite programming for spectral clustering: Modeling and approximation," Advanced Optimization Laboratory, McMaster University, Tech. Rep., 2005.
[37] J. Peng and Y. Wei, "Approximating k-means-type clustering via semidefinite programming," *SIOPT*, vol. 18, no. 1, 2007.
[38] E. P. Xing and M. I. Jordan, "On semidefinite relaxations for normalized k-cut and connections to spectral clustering," UC Berkeley, Tech. Rep. UCB/USD-3-1265, 2003.
[39] K. Lang, "Fixing two weaknesses of the spectral method," in *NIPS*, 2005, pp. 715–722.
[40] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, "Describing objects by their attributes," in *CVPR*, 2009, pp. 1778–1785.
[41] "Mulan: A Java Library for Multi-Label Learning," http://mulan.sourceforge.net/datasets.html.
[42] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognition*, vol. 37, 2004.
[43] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. P. Vlahavas, "Multi-label classification of music into emotions," in *ISMIR*, 2008.
[44] F. Pukelsheim, "The three sigma rule," *The American Statistician*, vol. 48, no. 2, pp. 88–91, 1994.
[45] A. Strehl and J. Ghosh, "Cluster ensembles – a knowledge reuse framework for combining multiple partitions," *JMLR*, 2003.
[46] J. Leskovec, "Stanford Network Analysis Project," http://snap.stanford.edu/.
[47] A. Benson, D. F. Gleich, and J. Leskovec, "Higher-order organization of complex networks," *Science*, vol. 353, pp. 163–166, 2016.

Joyce Jiyoung Whang is an assistant professor of Computer Science and Engineering at Sungkyunkwan University (SKKU) in Korea where she leads the Big Data Lab. She received her B.S. degree in Computer Science and Engineering from Ewha Womans University in Korea, and Ph.D. in Computer Science from the University of Texas at Austin. Her main research interests are in big data, data mining, network analysis, and machine learning with specific interests in community detection and overlapping clustering.



Yangyang Hou was a Ph.D. student and a postdoctoral research associate at Department of Computer Science, Purdue University. She now works with Ant Financial. She received Ph.D. degree from Purdue University and B.E. degree from Shanghai Jiao Tong University. Her research interests lie in matrix computation and optimization, especially with application to data mining, network analysis and machine learning.

David F. Gleich is an assistant professor of Computer Science at Purdue University. His research is on matrix computations, network and graph algorithms, and parallel and distributed computing. He received a Bachelor of Science degree from Harvey Mudd College, and a Ph.D. from Stanford University. He has been awarded a Microsoft Research Graduate fellowship, the John von Neumann postdoctoral fellowship, an NSF CAREER award, and a Sloan Research Fellowship.

Inderjit S. Dhillon is the Gottesman Family Centennial Professor of Computer Science and Mathematics at UT Austin, where he is also the Director of the ICES Center for Big Data Analytics. His main research interests are in big data, machine learning, network analysis, linear algebra and optimization. He received his B.Tech. degree from IIT Bombay, and Ph.D. from UC Berkeley. Inderjit has received several awards, including the ICES Distinguished Research Award, the SIAM Outstanding Paper Prize, the Moncrief Grand Challenge Award, the SIAM Linear Algebra Prize, the University Research Excellence Award, and the NSF Career Award. He has published over 175 journal and conference papers, and has served on the Editorial Board of the Journal of Machine Learning Research, the IEEE Transactions of Pattern Analysis and Machine Intelligence, Foundations and Trends in Machine Learning and the SIAM Journal for Matrix Analysis and Applications. Inderjit is an ACM Fellow, an IEEE Fellow, a SIAM Fellow and an AAAS Fellow.