Multiple Kernel Clustering with Dual Noise Minimization

Junpu Zhang*
Liang Li*
zhangjunpu@nudt.edu.cn
liangli@nudt.edu.cn
National University of Defense
Technology
Changsha, China

Siwei Wang
wangsiwei13@nudt.edu.cn
National University of Defense
Technology
Changsha, China

Jiyuan Liu
liujiyuan13@nudt.edu.cn
National University of Defense
Technology
Changsha, China

Yue Liu
yueliu@nudt.edu.cn
National University of Defense
Technology
Changsha, China

Xinwang Liu†
xinwangliu@nudt.edu.cn
National University of Defense
Technology
Changsha, China

En Zhu†
enzhu@nudt.edu.cn
National University of Defense
Technology
Changsha, China

ABSTRACT

Clustering is a representative unsupervised method widely applied in multi-modal and multi-view scenarios. Multiple kernel clustering (MKC) aims to group data by integrating complementary information from base kernels. As a representative, late fusion MKC first decomposes the kernels into orthogonal partition matrices, then learns a consensus one from them, achieving promising performance recently. However, these methods fail to consider the noise inside the partition matrix, preventing further improvement of clustering performance. We discover that the noise can be disassembled into separable dual parts, i.e. N-noise and C-noise (Null space noise and Column space noise). In this paper, we rigorously define dual noise and propose a novel parameter-free MKC algorithm by minimizing them. To solve the resultant optimization problem, we design an efficient two-step iterative strategy. To our best knowledge, it is the first time to investigate dual noise within the partition in the kernel space. We observe that dual noise will pollute the block diagonal structures and incur the degeneration of clustering performance, and C-noise exhibits stronger destruction than N-noise. Owing to our efficient mechanism to minimize dual noise, the proposed algorithm surpasses the recent methods by large margins.

CCS CONCEPTS

• **Computing methodologies → Cluster analysis**; • **Theory of computation → Unsupervised learning and clustering**.

---

*Both authors contributed equally to this research.
†Corresponding author.

KEYWORDS

Multi-view clustering, multiple kernel clustering, noise minimization

1 INTRODUCTION

Clustering is a representative unsupervised learning method widely applied in data mining, community detection and many other machine learning scenarios [1–7]. Multi-view or multi-modal clustering aims to optimally fuse diverse and complementary information, which has been a hotpot in current research [8–16]. As Figure 1 shows, how to effectively and efficiently integrate multimedia or multiple features, e.g. image, video, and text, is still an open question [17–24]. Multiple kernel clustering (MKC) [25–30] is a popular technique to solve this. Considering the insufficiency to tackle nonlinearly separable data in sample space, MKC maps the sample space to a Reproducing Kernel Hilbert Space (RKHS), where the data can be linearly separable [31]. Currently, there are two mainstream methods, including kernel fusion and late fusion strategies.

Kernel fusion methods focus on learning a consensus kernel from base kernels directly, afterwards compute the final partition (cluster soft-assignment) [32]. A typical paradigm is multiple kernel k-means (MKKM) [33]. Meanwhile, plenty of variants are derived [34–38]. For the purpose of directly serving for clustering tasks, late fusion methods aim to obtain a consensus partition from base partitions. This strategy is proposed by [39] and inspires a large number of researches [40–44]. Our proposed algorithm belongs to the second category.

Although the late fusion methods exhibit promising performance, most existing researches [39, 42–44] encounter three issues: (i) These models adopt a coarse manner that directly fuses the pre-computed kernel partitions without updating during the optimization. Consequently, the quality of consensus partition is greatly

Figure 1: Multi-view learning aims to fuse data across multi-media or multiple features, e.g. features can be extract from image, video, text and other multi-modal representations.
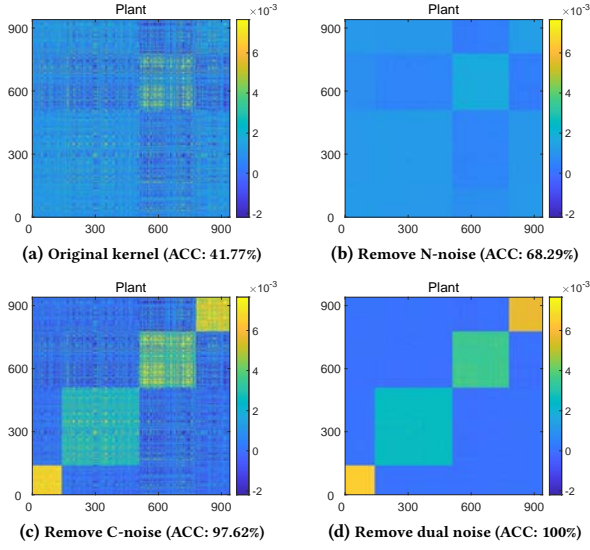

(a) Original kernel (ACC: 41.77%)

(b) Remove N-noise (ACC: 68.29%)

(c) Remove C-noise (ACC: 97.62%)

(d) Remove dual noise (ACC: 100%)

Figure 2: Visualization of noise destruction on Plant dataset.

limited by the initial partitions and leads to limited clustering performance. The work in [41] attempts to tackle this issue by updating partitions in a hierarchical manner. However, along with the improvement of clustering performance, it introduces a great complexity in the optimization. (ii) Moreover, most late fusion based methods are modeled with one or more hyper-parameters, which is intractable in real-world scenarios due to the missing of supervisory signals. (iii) Most critically, existing researches fail to consider the noise within partition matrices. In clustering settings, researchers always prefer a clear block diagonal structure. However, as Figure 2 (a) shows, the noise will inevitably corrupt the block diagonal structure, leading to the degeneration of clustering performance. Overall, an efficient, parameter-free model which can effectively minimize the impact of noise is an urgent need in multiple kernel clustering applications.

To fill these gaps, this paper develops a novel MKC algorithm with a dual noise minimization mechanism (MKC-DNM). Specifically, we discover that the noise, according to its mathematical property, can be disassembled into two separate dual parts, i.e. Null space noise (N-noise) and Column space noise (C-noise). As Figure 2 shows, we visualize the effect of dual noise on Plant dataset. Specifically, we test the kernel quality (the accuracy of kernel k-means) in four comparative settings, i.e. without modification, removing

N-noise, removing C-noise and removing both of them. It can be observed that the accuracy increases from 41.77% to 68.29% and 97.62% with removing N-noise and C-noise, respectively. The phenomenon illustrates (i) both of the dual noise will pollute the kernel, leading to the degeneration of clustering performance; (ii) C-noise exhibits stronger destruction than N-noise on the block diagonal structure. Therefore, a natural motivation of this work is to minimize dual noise. This paper firstly provides rigorous mathematical definitions of dual noise, then carefully explores their properties, and further proposes a unified and elegant paradigm to minimize them simultaneously. The contributions of this work are summarized as follows:

1) In MKC scenarios, for the first time, we mathematically disassemble the noise of kernel partition into N-noise and C-noise, distinguishing our work from existing researches. Furthermore, we find that C-noise exhibits stronger destruction than N-noise on the block diagonal structures, which directly leads to the degeneration of clustering performance.

2) We propose a novel model to minimize dual noise in late fusion framework. Most importantly, our model is parameter-free, making it practical, especially in unsupervised scenarios.

3) We propose an efficient two-step alternative optimization strategy to solve our model with linear computation complexity, and achieve state-of-art clustering performance on benchmark datasets.

2 RELATED WORK

2.1 Multiple Kernel k-means

Considering a data matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$ drawn from k clusters where d and n refer to the feature dimension and sample number respectively, k-means aims to minimize the inter-cluster loss [45–48],

$$\min_{\mathbf{Y}} \ \sum_{i=1}^{n} \sum_{r=1}^{k} \|\mathbf{x}_i - \mathbf{c}_r\|_2^2 \mathbf{Y}_{ir}, \quad \text{s.t.} \ \sum_{r=1}^{k} \mathbf{Y}_{ir} = 1, \tag{1}$$

where $\mathbf{Y} \in \{0, 1\}^{n \times k}$ is the indicator matrix, $n_r = \sum_{i=1}^{n} \mathbf{Y}_{ir}$ is the sample number of the r-th cluster whose centroid is $\mathbf{c}_r$.

With kernel trick [31], i.e. $\mathbf{K}_{ij} = \kappa\left(\mathbf{x}_i, \mathbf{x}_j\right) = \phi\left(\mathbf{x}_i\right)^\top \phi\left(\mathbf{x}_j\right)$, the sample space $\mathbb{R}^d$ can be mapped into an RKHS $\mathcal{H}$ [49], in which $\kappa\left(\cdot, \cdot\right)$ is the kernel function, $\phi(\cdot)$ is nonlinear feature mapping. Kernel k-means (KKM) is transformed to

$$\min_{\mathbf{H}} \ \text{Tr}\left(\left(\mathbf{I} - \mathbf{H}\mathbf{H}^\top\right)\mathbf{K}\right), \quad \text{s.t.} \ \mathbf{H}^\top\mathbf{H} = \mathbf{I}_k, \ \mathbf{H} \in \mathbb{R}^{n \times k}, \tag{2}$$

where the partition matrix $\mathbf{H}$ is computed by eigenvalue decomposition. The final cluster labels can be obtained by performing k-means on $\mathbf{H}$ [30].

In multiple kernel scenarios, the consensus kernel is commonly assumed as a combination of m base kernels. As a representative, the objective of MKKM [33] is

$$\min_{\mathbf{H}, \boldsymbol{\beta}} \ \text{Tr}\left(\left(\mathbf{I} - \mathbf{H}\mathbf{H}^\top\right)\mathbf{K}_{\boldsymbol{\beta}}\right),$$
$$\text{s.t.} \ \mathbf{H}^\top\mathbf{H} = \mathbf{I}_k, \ \mathbf{H} \in \mathbb{R}^{n \times k}, \ \boldsymbol{\beta}^\top\mathbf{1} = 1, \ \beta_p \geq 0, \ \forall p, \tag{3}$$

where $\mathbf{K}_{\boldsymbol{\beta}} = \sum_{p=1}^{m} \beta_p^2 \mathbf{K}_p$ is the consensus kernel and β_p is the weight of the p-th kernel. In the optimization, $\mathbf{H}$ and $\boldsymbol{\beta}$ can be solved alternatively.

2.2 Late Fusion Multiple Kernel Clustering

Instead of fusing consensus kernel from base kernels $\mathbf{K}_p$, late fusion MKC focuses on fusing multiple partitions $\mathbf{H}_p$ to directly serve for clustering [39]. The paradigm of late fusion MKC (LFMKC) is presented as follows:

$$
\begin{aligned}
\max_{\mathbf{H}^*, \mathbf{H}_{\boldsymbol{\beta}}, \boldsymbol{\beta}} \quad & \mathrm{Tr}\left(\mathbf{H}^{*\top}\mathbf{H}_{\boldsymbol{\beta}} + \lambda \mathbf{H}^{*\top}\mathbf{M}\right), \\
\text{s.t.} \quad & \mathbf{H}^{*\top}\mathbf{H}^* = \mathbf{I}_k, \ \mathbf{H}^* \in \mathbb{R}^{n \times k}, \\
& \mathbf{W}_p^\top \mathbf{W}_p = \mathbf{I}_k, \ \mathbf{W}_p \in \mathbb{R}^{k \times k}, \\
& \|\boldsymbol{\beta}\|_2^2 = 1, \beta_p \geq 0, \forall p,
\end{aligned}
\tag{4}
$$

where $\mathbf{H}^*$ is the consensus partition, $\mathbf{H}_{\boldsymbol{\beta}} = \sum_{p=1}^m \beta_p \mathbf{H}_p \mathbf{W}_p$ is the fused partition with each base one aligned by permutation matrix $\mathbf{W}_p$, $\mathbf{M}$ is the partition of average kernel, and λ is a trade-off parameter.

The above paradigm aims to maximally align the consensus partition and base partitions. Although achieving promising performance, the consensus partition, as pointed in [41], is directly learned from base partitions that are fixed during the optimization, limiting its performance. Moreover, the current method neglects the noise in kernel partitions.

2.3 Hierarchical Multiple Kernel Clustering

To update the kernel partition during optimization, [41] proposes to gradually category clusters from $\mathbf{K}_p \in \mathbb{R}^{n \times n}$ to intermediate $\mathbf{H} \in \mathbb{R}^{n \times c}$ and finally to $\mathbf{H}^* \in \mathbb{R}^{n \times k}$. The idea is formulated as

$$
\begin{aligned}
\max_{\mathbf{H}^*, \mathbf{H}_p, \boldsymbol{\omega}, \boldsymbol{\beta}} \quad & \sum_{t=1}^{s}\sum_{p=1}^{m} \omega_p^{(t)} \mathrm{Tr}\left(\mathbf{K}_p^{(t)}\mathbf{K}_p^{(t-1)}\right) + \sum_{p=1}^{m} \beta_p \mathrm{Tr}\left(\mathbf{K}_p^{(s)}\mathbf{K}^*\right), \\
\text{s.t.} \quad & \mathbf{H}^{*\top}\mathbf{H}^* = \mathbf{I}_k, \ \mathbf{H}_p^{(t)\top}\mathbf{H}_p^{(t)} = \mathbf{I}_{c_t}, \ \mathbf{H}^* \in \mathbb{R}^{n \times k}, \\
& \mathbf{H}_p^{(t)} \in \mathbb{R}^{n \times c_t}, \ n > c_1 > \cdots > c_s > k, \\
& \|\boldsymbol{\omega}^{(t)}\|_2^2 = 1, \ \omega_p^{(t)} \geq 0, \ \|\boldsymbol{\beta}\|_2^2 = 1, \ \beta_p \geq 0,
\end{aligned}
\tag{5}
$$

where $\mathbf{K}_p^{(t)} = \mathbf{H}_p^{(t)}\mathbf{H}_p^{(t)\top}$ for $t \geq 1$, $\mathbf{K}_p^{(0)} = \mathbf{K}_p$, $\mathbf{K}^* = \mathbf{H}^*\mathbf{H}^{*\top}$, and $\mathbf{H}_p^{(t)}$ is the intermediary partitions with decreasing sizes. The complex formulation conveys a straightforward insight that the data representation should be extracted step by step. In this way, the data information beneficial to clustering could be maximally preserved.

Obviously, each term of Eq. (5) is a kernel k-means objective in essence. Also, this is an empirical and coarse manner. Nevertheless, the sizes of intermediary partitions are still hyper-parameters that require carefully tuning or grid-searching in practice. Most importantly, it fails to tackle the noise inside the partition matrices.

3 METHODOLOGY

3.1 Motivation

In multiple kernel scenarios, a d_p-dimensional feature matrix $\mathbf{U}_p \in \mathbb{R}^{n \times d_p}$ is commonly served as the data representation of the p-th kernel computed by singular value decomposition (SVD), which satisfies $\mathbf{U}_p^\top \mathbf{U}_p = \mathbf{I}_{d_p}$. It's worth noting that the dimension d_p of

feature matrices may vary at a large range since multiple kernels are naturally discrepant and complementary. Consequently, it is necessary to fuse a consensus optimal partition $\mathbf{H}^* \in \mathbb{R}^{n \times k}$ across multiple $\mathbf{U}_p$ for clustering purpose, i.e.

$$
f\left(\{\mathbf{U}_p\}_{p=1}^m\right) = \mathbf{H}^*, \ \ \text{s.t.} \ \mathbf{H}^{*\top}\mathbf{H}^* = \mathbf{I}_k,
\tag{6}
$$

where f is the function to fuse m feature matrices.

3.1.1 Definitions of dual noise. Directly integrating feature matrices $\mathbf{U}_p$ across m kernels is a challenging issue since their dimensions d_p varies greatly. Fortunately, both $\mathbf{U}_p\mathbf{U}_p^\top$ and $\mathbf{H}^*\mathbf{H}^{*\top}$ share the target clustering structure of $\mathbf{K}_p$ but with a discrepancy $\mathbf{E}_p$, i.e.

$$
\mathbf{U}_p\mathbf{U}_p^\top = \mathbf{H}^*\mathbf{H}^{*\top} + \mathbf{E}_p, \ \ \forall p \in \Delta_m,
\tag{7}
$$

where $\Delta_m = \{1, 2, \cdots, m\}$, and $\mathbf{E}_p \in \mathbb{R}^{n \times n}$ can be regarded as the noise matrix. Mathematically, $\mathbf{E}_p$ can be further separated into Null space noise (N-noise, $\mathbf{E}_p^\mathrm{N}$) and Column space noise (C-noise, $\mathbf{E}_p^\mathrm{N}$), i.e.

$$
\mathbf{E}_p = \mathbf{E}_p^\mathrm{N} + \mathbf{E}_p^\mathrm{C}, \ \ \forall p \in \Delta_m.
\tag{8}
$$

We emphasize that the above definitions are derived based on which subspace their eigenvectors belong to, i.e. $v\left(\mathbf{E}_p^\mathrm{N}\right) \in N\left(\mathbf{H}^{*\top}\right)$, $v\left(\mathbf{E}_p^\mathrm{C}\right) \in C\left(\mathbf{H}^*\right)$, where $v\left(\mathbf{A}\right)$ denotes the eigenvectors with corresponding non-zero eigenvalues of matrix $\mathbf{A}$, $N\left(\mathbf{B}\right)$ and $C\left(\mathbf{B}\right)$ denote the Null space and the Column space of matrix $\mathbf{B}$, respectively. Mathematically, as pointed in [50], $v\left(\mathbf{E}_p^\mathrm{N}\right)$, $v\left(\mathbf{E}_p^\mathrm{C}\right)$ can be computed by

$$
v\left(\mathbf{E}_p^\mathrm{N}\right) \in N\left(\mathbf{H}^{*\top}\right) = \left\{\mathbf{x} \mid \mathbf{H}^{*\top}\mathbf{x} = \mathbf{0}\right\}, \ \ \forall p \in \Delta_m,
\tag{9}
$$

$$
v\left(\mathbf{E}_p^\mathrm{C}\right) \in C\left(\mathbf{H}^*\right) = \left\{\mathbf{y} \mid \exists \, \mathbf{x}, \ \text{s.t.} \ \mathbf{y} = \mathbf{H}^*\mathbf{x}\right\}, \ \ \forall p \in \Delta_m.
\tag{10}
$$

According to [50], the null space of $\mathbf{H}^{*\top}$ is the orthogonal complement of the column space of $\mathbf{H}^*$, which demonstrates that given $\mathbf{E}_p$, dual noise matrices $\mathbf{E}_p^\mathrm{N}$ and $\mathbf{E}_p^\mathrm{C}$ exist and should be unique. Figure 2 gives a visualization of destruction caused by dual noise in kernel space.

3.1.2 Properties of dual noise. Before introducing the proposed noise minimization mechanism, we first give several vital mathematical properties of $\mathbf{E}_p^\mathrm{N}$ and $\mathbf{E}_p^\mathrm{C}$ in Lemma 3.1-3.3.

LEMMA 3.1. $\mathrm{Tr}\left(\mathbf{E}_p^\mathrm{N}\mathbf{H}^*\mathbf{H}^{*\top}\right) = 0, \ \ \forall p \in \Delta_m.$

LEMMA 3.2. $\mathrm{Tr}\left(\mathbf{E}_p^\mathrm{C}\mathbf{H}^*\mathbf{H}^{*\top}\right) = \mathrm{Tr}\left(\mathbf{E}_p^\mathrm{C}\right), \ \ \forall p \in \Delta_m.$

LEMMA 3.3. $\mathbf{E}_p^\mathrm{N}$ *is positive semi-definite (PSD) and* $\mathbf{E}_p^\mathrm{C}$ *is negative semi-definite (NSD).*

Lemma 3.1-3.2 illustrate the relationship between the dual noise and the optimal partition $\mathbf{H}^*$, and Lemma 3.3 shows that we can remove dual noise by minimizing the absolute value of their trace. Due to space limit, the detailed proofs of Lemma 3.1-3.3 are provided in the appendix.
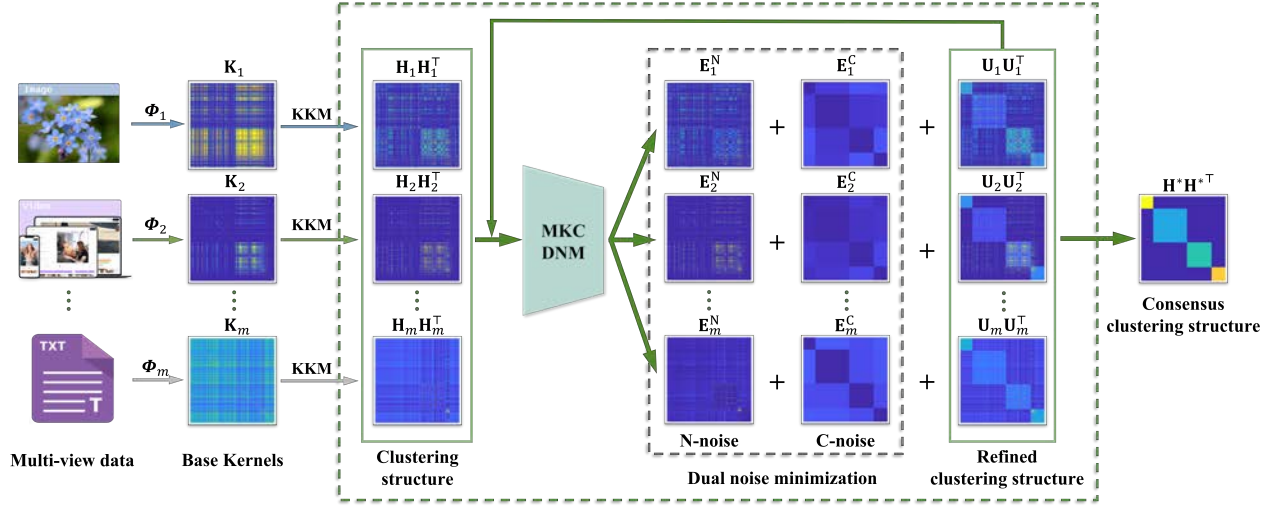
Figure 3: The framework of the proposed MKC-DNM model. The core idea is to adaptively optimize base partitions by minimizing dual noise during the iteration. Specifically, the MKC-DNM model firstly performs kernel k-means and generate $\mathbf{H}_p\mathbf{H}_p^\top$ to recover the clustering structures of base kernels K_p, and then gradually remove N-noise ($\mathbf{E}_p^N$) and C-noise ($\mathbf{E}_p^C$) of $\mathbf{H}_p$, preserving the denoised feature matrices $\mathbf{U}_p$. Consequently, a consensus partition $\mathbf{H}^*$ with precise block diagonal structures is obtained.

3.1.3 Minimizing C-noise. Recall our motivation that we aim to minimize C-noise, ideally $\mathbf{E}_p^C = 0$, according to Lemma 3.2 and Lemma 3.3, it is equivalent to

$$\text{Tr}\left(\mathbf{E}_p^C \mathbf{H}^*\mathbf{H}^{*\top}\right) = 0, \quad \forall p \in \Delta_m. \tag{11}$$

However, directly solving Eq. (11) is difficult due to the unknown $\mathbf{H}^*$. Fortunately, Theorem 3.1 provides a necessary condition to satisfy $\mathbf{E}_p^C = 0$.

Theorem 3.1. $\|\mathbf{U}_p^\top\mathbf{U}_q\|_F^2 \geq k$ *is necessary for* $\mathbf{E}_p^C = 0$, $\forall p \in \Delta_m$.

Proof. Given $\mathbf{E}_p^C = 0$ and $\mathbf{H}^{*\top}\mathbf{H}^* = \mathbf{I}_k$, according to Eq. (6), we have

$$\|\mathbf{U}_p^\top\mathbf{U}_q\|_F^2 = \text{Tr}\left(\mathbf{U}_p\mathbf{U}_p^\top\mathbf{U}_q\mathbf{U}_q^\top\right)$$
$$= \text{Tr}\left(\left(\mathbf{H}^*\mathbf{H}^{*\top} + \mathbf{E}_p^N\right)\left(\mathbf{H}^*\mathbf{H}^{*\top} + \mathbf{E}_q^N\right)\right) \tag{12}$$
$$= \text{Tr}\left(\mathbf{H}^*\mathbf{H}^{*\top}\mathbf{H}^*\mathbf{H}^{*\top} + \mathbf{E}_p^N\mathbf{H}^*\mathbf{H}^{*\top} + \mathbf{H}^*\mathbf{H}^{*\top}\mathbf{E}_q^N + \mathbf{E}_p^N\mathbf{E}_q^N\right).$$

According to Lemma (3.1), Eq. (12) is equivalent to

$$\text{Tr}\left(\mathbf{H}^*\mathbf{H}^{*\top}\mathbf{H}^*\mathbf{H}^{*\top} + \mathbf{E}_p^N\mathbf{E}_q^N\right) = k + \text{Tr}\left(\mathbf{E}_p^N\mathbf{E}_q^N\right). \tag{13}$$

Since $\mathbf{E}_p^N$ satisfies PSD according to Lemma 3.3, we have $\text{Tr}\left(\mathbf{E}_p^N\mathbf{E}_q^N\right) \geq 0$, i.e. Eq. (13) $\geq k$. Consequently, $\|\mathbf{U}_p^\top\mathbf{U}_q\|_F^2 \geq k$.
This completes the proof. □

3.1.4 Minimizing N-noise. Similarly, the optimal solution of minimizing N-noise is $\mathbf{E}_p^N = \mathbf{0}$. According to Lemma 3.3, it is equivalent to $\text{Tr}(\mathbf{E}_p^N) = 0$. Therefore, the original goal to minimize $\mathbf{E}_p^N$ can be transformed to minimize $\text{Tr}(\mathbf{E}_p^N)$. For MKC scenario, we should minimize $\sum_{p=1}^m \text{Tr}(\mathbf{E}_p^N)$. Furthermore, Theorem 3.2 illustrates that minimizing $\sum_{p=1}^m \text{Tr}(\mathbf{E}_p^N)$ is equivalent to minimizing $\mathbf{d}^\top\mathbf{1}$.

Theorem 3.2. *Fixing* $\{\mathbf{E}_p^C\}_{p=1}^m$, *minimizing* $\sum_{p=1}^m \text{Tr}(\mathbf{E}_p^N)$ *is equivalent to minimizing* $\mathbf{d}^\top\mathbf{1}$.

Proof. Given $\mathbf{U}_p\mathbf{U}_p^\top = \mathbf{I}_{d_p}$ and $\mathbf{H}^{*\top}\mathbf{H}^* = \mathbf{I}_k$ for all $p \in \Delta_m$, according to Eq. (7), we have

$$\sum_{p=1}^m \text{Tr}\left(\mathbf{E}_p^N\right) = \sum_{p=1}^m \left(\text{Tr}\left(\mathbf{U}_p\mathbf{U}_p^\top\right) - \text{Tr}\left(\mathbf{H}^*\mathbf{H}^{*\top} + \mathbf{E}_p^C\right)\right)$$
$$= \sum_{p=1}^m d_p - \sum_{p=1}^m \left(k + \text{Tr}\left(\mathbf{E}_p^C\right)\right) \iff \sum_{p=1}^m d_p = \mathbf{d}^\top\mathbf{1}, \tag{14}$$

where $\mathbf{d} = [d_1, d_2, \cdots, d_m]^\top$ collects the dimension of feature matrices $\mathbf{U}_p$. Note that the above deduction is valid since $\sum_{p=1}^m \left(k + \text{Tr}\left(\mathbf{E}_p^C\right)\right)$ is a constant and can be eliminated for optimization.
This completes the proof. □

3.2 Proposed Formulation

According to the aforementioned analysis on minimizing dual noise, we integrate Theorem 3.1 and Theorem 3.2 into a unified and parameter-free framework, i,e.

$$\min_{\mathbf{d}} \mathbf{d}^\top\mathbf{1}, \quad \text{s.t. } \|\mathbf{U}_p^\top\mathbf{U}_q\|_F^2 \geq k, \ \forall p, q \in \Delta_m; \ \mathbf{d} \in \mathbb{Z}_+^m, \tag{15}$$

where $\mathbf{d} = [d_1, d_2, \cdots, d_m]^\top$ collects the dimensions of feature matrices $\mathbf{U}_p$, and $\mathbb{Z}_+^m$ denotes the set of all positive integers.

From the above compact model, we have the following observations: (i) Our motivation aims to minimize dual noise, and we derive a straightforward but elegant framework. (ii) The objective is derived from minimizing N-noise, and the constraint is originated from minimizing C-noise, which directly serves for noise minimization purpose. (iii) Our model is parameter-free, satisfying the essence of unsupervised clustering.

Algorithm 1 Multiple kernel clustering with dual noise minimization

Input: $\{H_p\}_{p=1}^m$ and k.
Output: The optimal $\{U_p\}_{p=1}^m$ and $\mathbf{d}$.

1: Initialize a feasible $\mathbf{d}$, $\hat{\mathbf{d}} = \mathbf{d}$, flag = 1;
2: **while** flag **do**
3:    update $\mathbf{d}$ by optimizing Eq. (22);
4:    update $\hat{\mathbf{d}}$ by optimizing Eq. (23);
5:    **if** $\mathbf{d} = \hat{\mathbf{d}}$ **then**
6:       flag = 0.
7:    **end if**
8: **end while**

Although our algorithm is compact and elegant, it is difficult to solve Eq. (15) directly. Inspired by the widely employed Big M method [51] in operation research, we introduce auxiliary variables $\{a_p\}_{p=1}^m$ to transform the original Eq. (15) into the following formulation:

$$
\min_{\mathbf{d},\hat{\mathbf{d}}} \quad \frac{1}{2}(\mathbf{d}+\hat{\mathbf{d}})^\top \mathbf{1} + \frac{M}{2}\|\mathbf{d}-\hat{\mathbf{d}}\|_2^2,
$$
$$
\text{s.t.} \quad \|U_p^\top \hat{U}_q\|_F^2 \geq k, \ \forall p,q \in \Delta_m; \ \mathbf{d},\hat{\mathbf{d}} \in \mathbb{Z}_+^m, \tag{16}
$$

where $\hat{\mathbf{d}} = \mathbf{d} + \mathbf{a}$ and $\hat{U}_q \in \mathbb{R}^{n \times \hat{d}_p}$. As pointed in [51], the solution of Eq. (15) is equivalent to Eq. (16) for a large enough M, i.e. $\|\mathbf{d}-\hat{\mathbf{d}}\|_2^2$ will be zero. Theorem 3.3 gives a brief proof.

Theorem 3.3. *The local optimal solution of Eq. (16) is equivalent to that of Eq. (15) for a large enough constant M.*

Proof. We first define two functions:

$$
g(\mathbf{d}) = \mathbf{d}^\top \mathbf{1}, \quad G_M(\mathbf{d},\hat{\mathbf{d}}) = \frac{1}{2}(\mathbf{d}+\hat{\mathbf{d}})^\top \mathbf{1} + \frac{M}{2}\|\mathbf{d}-\hat{\mathbf{d}}\|_2^2, \tag{17}
$$

Supposing $\left(\mathbf{d}^*, \hat{\mathbf{d}}^*\right)$ is a local optimal solution of Eq. (16), which can be formulated as

$$
\exists N\left(\mathbf{d}^*, \hat{\mathbf{d}}^*\right), \forall \left(\mathbf{d}, \hat{\mathbf{d}}\right) \in N\left(\mathbf{d}^*, \hat{\mathbf{d}}^*\right), \ G_M\left(\mathbf{d}^*, \hat{\mathbf{d}}^*\right) \leq G_M\left(\mathbf{d}, \hat{\mathbf{d}}\right), \tag{18}
$$

where $N(\bullet)$ denotes the neighborhood of $\bullet$.

A large enough M means $\|\mathbf{d}^* - \hat{\mathbf{d}}^*\|_2^2 = 0$, i.e. $\mathbf{d}^* = \hat{\mathbf{d}}^*$. According to Eq. (18), we have

$$
\exists N\left(\mathbf{d}^*\right), \forall \mathbf{d} \in N\left(\mathbf{d}^*\right), \ G_M\left(\mathbf{d}^*, \mathbf{d}^*\right) \leq G_M\left(\mathbf{d}, \mathbf{d}\right). \tag{19}
$$

Furthermore, we have

$$
g\left(\mathbf{d}^*\right) = G_M\left(\mathbf{d}^*, \mathbf{d}^*\right) \leq G_M\left(\mathbf{d}, \mathbf{d}\right) = g\left(\mathbf{d}\right). \tag{20}
$$

This completes the proof. □

3.3 Optimization

Directly optimizing Eq. (16) is difficult since it is not convex. This section provides an alternative strategy where each sub-optimization is convex.

3.3.1 Sub-optimization of updating $\mathbf{d}$. With fixed $\hat{\mathbf{d}}$, Eq. (16) is formulated into

$$
\min_{\{d_p\}_{p=1}^m} \sum_{p=1}^m \frac{1}{2}(d_p + \hat{d}_p) + \frac{M}{2}(d_p - \hat{d}_p)^2,
$$
$$
\text{s.t.} \quad \|U_p^\top \hat{U}_q\|_F^2 \geq k, \ \forall q \in \Delta_m; \ d_p \in \mathbb{Z}_+. \tag{21}
$$

As pointed in [52], the object function of Eq. (21) is $L^\natural$-convex on the effective domain, and the effective domain is restricted to an $L^\natural$-convex set. Consequently, Eq. (21) is an $L^\natural$-convex problem about $\mathbf{d}$.

Since each d_p is independent, Eq. (21) can be transformed into m independent sub-problems and each can be separately solved by

$$
\min_{d_p} \quad \frac{1}{2}(d_p + \hat{d}_p) + \frac{M}{2}(d_p - \hat{d}_p)^2,
$$
$$
\text{s.t.} \quad \|U_p^\top \hat{U}_q\|_F^2 \geq k, \ \forall q \in \Delta_m, \ d_p \in \mathbb{Z}_+. \tag{22}
$$

According to [53], the above one-dimensional $L^\natural$-convex optimization problem can be solved by discrete binary search efficiently.

3.3.2 Sub-optimization of updating $\hat{d}$. Similarly, we can obtain $\hat{\mathbf{d}}$ by solving the following problem separately for each p:

$$
\min_{\hat{d}_p} \quad \frac{1}{2}(\hat{d}_p + d_p) + \frac{M}{2}(\hat{d}_p - d_p)^2,
$$
$$
\text{s.t.} \quad \|\hat{U}_p^\top U_q\|_F^2 \geq k, \ \forall q \in \Delta_m, \ \hat{d}_p \in \mathbb{Z}_+. \tag{23}
$$

The optimization procedures in solving Eq. (16) is outlined in Algorithm 1. Note that the optimal $U_p U_p^\top$ learned by Algorithm 1 is PSD for all $p \in \Delta_m$, which can be regarded as kernel matrix. As a result, we can employ average kernel k-means on $\{U_p U_p^\top\}_{p=1}^m$ to recover the optimal H^*, which can be solved efficiently by performing SVD on $[U_1, U_2, \cdots, U_m]$ and extract the rank-k columns of the left singular matrix [54].

3.4 Complexity and Convergence

3.4.1 Computational Complexity. The computation complexity of our algorithm involves two part, i.e. optimization and post-processing. In optimization, it involves two variables. Updating $\mathbf{d}$ involves m independent sub-problems to compute the optimal $\mathbf{d}$, and each one conducts discrete binary search. Therefore, the computational complexity is $O\left(nm^2k^2\log k\right)$. Similarly, updating $\hat{\mathbf{d}}$ also needs $O\left(nm^2k^2\log k\right)$ for solving Eq. (23). Therefore, the optimization process costs $O\left(nm^2k^2\log k\right)$. Note that it achieves linear complexity with respect to sample number. For post-processing, we perform SVD on $[U_1, U_2, \cdots, U_m]$ to obtain the optimal partition matrix H^* and get the final clustering label by k-means. The complexity is $O\left(nm^2k^2\right)$, which is also linear complexity with respect to n. Consequently, we develop a linear-complexity algorithm with $O\left(nm^2k^2\log k\right)$, sharing the similar computational complexity with late fusion methods [39] and [42]. Note that linear complexity means it is suitable for large-scale tasks. Moreover, our algorithm is free of hyper-parameter, which is more practical compared with the ones requiring parameter-tuning by grid search.

3.4.2 Convergence. Our algorithm is non-convex to directly compute all the variables, and we adopt an alternative optimization manner to solve the resultant model. According to Theorem 1 in

Table 1: Information of datasets

Dataset	Type	Samples	Views	Clusters
BBCSport	Text	544	2	5
Plant	Image	940	69	4
SensIT Vehicle	Graph	1500	2	3
CCV	Video	6773	3	20
Flower102	Image	8189	4	102
Reuters	Text	18758	5	6

[55], alternatively optimizing each sub-optimization is convex with other variables fixed. The objective of Algorithm 1 is monotonically decreased and lower bounded by zero. Consequently, our proposed algorithm can be guaranteed convergent to a local optimal solution.

4 EXPERIMENT

4.1 Datasets

Table 1 lists six MKC datasets, including BBCSport [1], Plant[2], SensIT Vehicle[3], CCV[4], Flower102[5], and Reuters[6]. The datasets vary in type and size, which will provide convincing evaluation for this work. All the datasets are downloaded form public websites.

4.2 Compared Methods

Ten existing graph or kernel based multi-view clustering (MVC) models are adopted as baseline, i.e.
(1) **AKKM** conducts KKM in average kernel space.
(2) **MKKM** [33] combines base kernels with learned weights.
(3) **MKKM-MR** [34] proposes a matrix-induced regularization to avoid the sparsity of weight.
(4) **SWMC** [56] is a self-weighted method.
(5) **ONKC** [36] learns the optimal neighborhood kernel.
(6) **LFMKC** [39] maximizes the alignment of permuted base partitions and the consensus one.
(7) **SPMKC** [57] simultaneously extract the global and local clustering structures by graph learning in MVC.
(8) **CAGL** [58] fuses a consensus graph across multiple kernels by graph learning with rank constraint.
(9) **OPLF** [42] is a one-pass version of LFMKC.
(10) **HKMC** [41] reduces the dimension of data representation hierarchically rather than abruptly.

Note that (i) the kernel fusion MKC algorithms, including AKKM, MKKM, MKKM-MR, and ONKC, (ii) graph learning based method SWMC, (iii) graph based MKC models, i.e. SPMKC and CAGL, share the similar computational complexity, i.e. $O(n^3)$. (iv) late fusion based algorithms, i.e. LFMKC and OPLF, are of $O(n)$ complexity. Since HKMC involves eigenvalue decomposition during the optimization, it costs $O(n^3)$ complexity.

---

[1] http://mlg.ucd.ie/datasets/bbc.html

[2] https://bmi.inf.ethz.ch/supplements/protsubloc

[3] https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/multiclass.html

[4] https://www.ee.columbia.edu/ln/dvmm/CCV/

[5] http://www.robots.ox.ac.uk/~vgg/data/flowers/

[6] http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html

4.3 Experimental Setup

For the MKC datasets, we suppose the real clusters k is pre-knowledge. All the public datasets are centered and normalized by following [31]. For the methods involving k-means, the cluster centroids are initialized 50 times randomly and we report the average results, reducing the randomness. For the compared methods, the involved hyper-parameters of their public codes are carefully tuned according to authors' settings. For the proposed MKC-DNM model, the involved parameter M in the optimization is determined in the following way to accelerate the procedure: M is firstly set with a small value, then in each iteration, we multiply M by two until satisfying the convergence condition, i.e. $\mathbf{d} = \hat{\mathbf{d}}$. The clustering performance is analyzed by four metrics, i.e. Accuracy (ACC), Normalized Mutual Information (NMI), Purity, and Adjusted Rand Index (ARI) [59–63]. The experiments are conducted on a computer with Intel Core i9 10900X CPU (3.5GHz), 64GB RAM, and MATLAB 2020b (64bit).

4.4 Clustering Performance

Table 2 lists four metrics comparison. The best results are marked in bold, the second best ones are marked in italic underline, and 'N/A' means unavailable results due to out of memory or time-out errors. Moreover, we record the average rank for a reference. From the reported results, we have the following observations:

1) The proposed algorithm exhibits the best or second best performance on six datasets. Compared to the late fusion based MKC methods, i.e. LFMKC, OPLF, and HMKC methods, our algorithm exhibits obvious improvement on most datasets, owing to our noise minimization mechanism.

2) Especially, HMKC can be considered as the most powerful competitor in MKC, our model still achieves comparable and better solutions on most datasets, with an obvious margin of 15.71%, 2.59%, 5.93%, 5.32%, 1.73%, and 1.33% of ACC on six datasets. We emphasize that our model is parameter-free in contrast to HMKC which introduces two empirical hyper-parameters and reports the best solution.

3) According to the average rank, our algorithm achieves the first rank over HMKC, LFMKC and OPLF methods, illustrating the superiority of our MKC-DNM algorithm.

4.5 Computation Time Comparison

Table 3 records the comparison of CPU time, including 'Tune Time' and 'Run Time'. 'Tune Time' records the execution time including hyper-parameter tuning. 'Run Time' records single running time. '-' denotes a parameter-free model. 'N/A' denotes unavailable results due to memory-out or time-out errors. Note that OPLF is parameter-free but requires 20 times computation to reduce the randomness of k-means. From the results, we observe that:

1) Compared to late fusion based LFMKC, OPLF, and HMKC, our model achieves comparable or shorter 'Run Time' than LFMKC and OPLF on most datasets, validating they share linear computational complexity. Not surprisingly, our model requires much less time than HMKC which has $O(n^3)$ complexity. Since the compared ones involve hyper-parameters or repeatably computation, they need much more 'Tune Time' than our MKC-DNM.

2) Compared to parameter-free kernel fusion based AKKM and MKKM with $O(n^3)$ complexity, although our model with $O(n)$

Table 2: Comparing the ACC, NMI, purity and ARI of eleven algorithms on six MVC datasets. The best one is marked in bold, the second best is marked in italic underline. 'N/A' denotes out of memory and time-out errors. 'Average Rank' records the average rank across six datasets.

Datasets	AKKM	MKKM [33]	MKKM-MR [34]	SWMC [56]	ONKC [36]	LFMKC [39]	SPMKC [57]	CAGL [58]	OPLF [42]	HMKC [41]	Proposed
Number of parameter	-	-	1	-	2	1	2	2	-	2	-
ACC (%)											
BBCSport	39.47	39.38	39.51	36.76	39.71	60.06	46.51	71.51	60.85	*71.90*	**87.61**
Plant	*61.28*	56.05	50.27	38.94	41.43	59.53	32.87	43.09	48.51	59.21	**61.79**
SensIT Vehicle	53.73	53.36	54.13	34.67	54.21	66.28	54.20	34.40	54.87	*66.60*	**72.53**
CCV	19.63	17.99	21.24	10.84	22.39	25.13	9.67	12.58	22.74	*33.31*	**35.04**
Flower102	27.07	22.41	40.22	6.72	39.55	38.45	N/A	26.25	29.78	**48.21**	*47.93*
Reuters	45.46	45.44	46.15	27.12	41.85	45.68	N/A	N/A	44.65	*46.84*	**48.17**
Average Rank	6.33	7.83	5.50	10.00	6.33	3.83	8.50	7.80	5.33	2.17	**1.17**
NMI (%)											
BBCSport	15.74	15.69	15.77	2.63	16.10	40.38	23.89	**72.74**	41.46	50.50	*69.70*
Plant	26.53	19.49	20.37	0.51	10.49	23.35	0.21	11.90	13.67	*27.98*	**31.06**
SensIT Vehicle	10.83	10.25	11.32	1.55	11.31	*23.53*	20.31	1.45	12.28	22.08	**32.09**
CCV	16.84	15.04	18.03	1.07	18.52	20.09	1.60	6.08	18.72	*29.85*	**30.79**
Flower102	46.02	42.67	56.71	5.51	56.11	54.94	N/A	45.09	46.77	**61.92**	*61.73*
Reuters	27.37	27.35	25.30	1.35	22.27	27.39	N/A	N/A	27.09	**31.04**	*30.64*
Average Rank	6.33	7.83	5.83	10.33	6.67	3.67	7.75	7.40	5.33	2.00	**1.50**
Purity (%)											
BBCSport	48.89	48.86	48.91	37.50	49.13	68.78	52.39	73.90	68.57	*74.65*	**87.61**
Plant	*61.28*	56.05	56.71	39.36	49.02	59.53	39.15	46.81	52.87	59.77	**62.10**
SensIT Vehicle	53.73	53.36	54.13	35.13	54.21	66.28	54.20	34.40	54.87	*66.60*	**72.53**
CCV	23.75	22.18	23.74	11.35	24.64	28.16	11.78	13.58	26.52	*37.05*	**38.32**
Flower102	32.27	27.79	46.34	8.08	45.63	44.56	N/A	31.33	34.03	*54.81*	**54.93**
Reuters	53.01	52.94	52.15	28.25	52.63	53.23	N/A	N/A	52.92	*53.91*	**55.37**
Average Rank	6.00	7.83	6.33	10.17	6.00	3.67	8.25	8.00	5.33	2.17	**1.00**
ARI (%)											
BBCSport	9.28	9.25	9.31	0.47	9.63	31.57	14.30	*56.64*	30.90	44.94	**69.60**
Plant	*24.64*	17.42	19.03	-0.25	9.78	21.66	-0.39	1.76	12.28	24.41	**28.90**
SensIT Vehicle	10.84	10.29	11.33	0.09	11.36	25.32	13.99	0.03	13.57	*25.55*	**35.51**
CCV	6.60	5.75	7.15	-0.02	7.74	9.44	-0.01	0.29	8.02	*14.74*	**15.70**
Flower102	15.46	12.06	25.49	0.10	24.86	25.46	N/A	2.28	18.92	**34.34**	*34.23*
Reuters	21.84	21.82	**23.08**	0.09	20.32	22.10	N/A	N/A	21.25	22.59	*22.72*
Average Rank	6.33	7.83	5.00	10.33	6.50	3.67	7.75	8.00	5.67	2.33	**1.33**

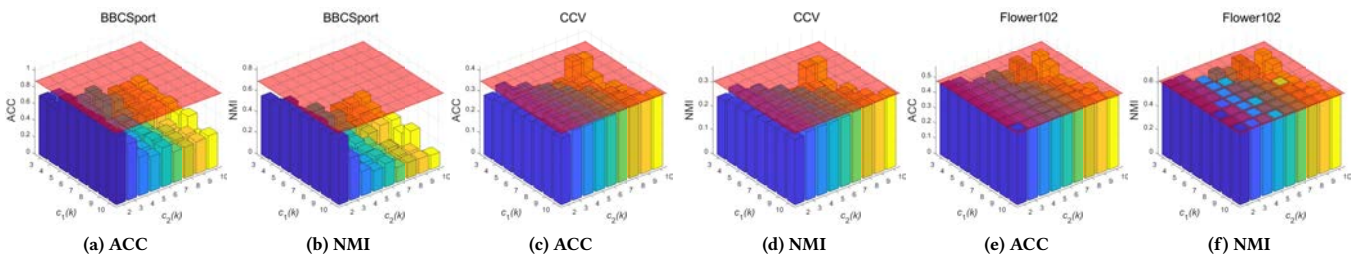(a) ACC (b) NMI (c) ACC (d) NMI (e) ACC (f) NMI

Figure 4: Comparing the strongest competitor HMKC and our algorithm on BBCSport, CCV and Flower102. The red slice denotes the performance of our parameter-free MKC-DNM. The 3-D bar denotes the results of HKMC which involves two hyper-parameters tuning by grid search.

Table 3: Comparing the CPU time. 'Tune Time' denotes execution time including hyper-parameter tuning. 'Run Time' denotes running time for this algorithm. '-' denotes a parameter-free algorithm. 'N/A' denotes out of memory and time-out errors.

Datasets		AKKM	MKKM	MKKM-MR	SWMC	ONKC	LFMKC	SPMKC	CAGL	OPLF	HMKC	Proposed
BBCSport	Tune Time	-	-	2.17	-	155.13	0.74	28.88	185.68	1.78	27.04	-
	Run Time	0.04	0.04	0.07	7.17	0.16	0.02	0.80	1.15	0.09	0.38	0.02
Plant	Tune Time	-	-	1032.42	-	31740.88	25.19	1119.20	9813.97	21.04	1542.74	-
	Run Time	0.16	3.88	33.30	101.68	33.03	0.81	31.09	60.58	1.05	21.43	15.29
SensIT Vehicle	Tune Time	-	-	16.63	-	1734.24	3.28	441.30	1532.34	3.41	127.52	-
	Run Time	0.04	0.56	0.54	57.23	1.80	0.11	12.26	9.46	0.17	1.77	0.22
CCV	Tune Time	-	-	1598.44	-	347868.59	208.92	364927.26	181525.46	148.10	21775.50	-
	Run Time	1.64	42.67	51.56	5512.67	361.99	6.74	10136.87	1120.53	7.40	302.44	55.99
Flower102	Tune Time	-	-	4343.86	-	624586.09	1195.65	N/A	382874.57	973.52	395255.46	-
	Run Time	7.27	109.76	140.12	7612.89	649.93	38.57	N/A	2363.42	48.68	5489.66	237.29
Reuters	Tune Time	-	-	49510.69	-	3743171.77	1571.38	N/A	N/A	983.08	93400.45	-
	Run Time	11.43	830.46	1597.12	84059.26	3895.08	50.69	N/A	N/A	49.15	1297.23	44.57

(a) Plant

(b) CCV

(c) Flower102
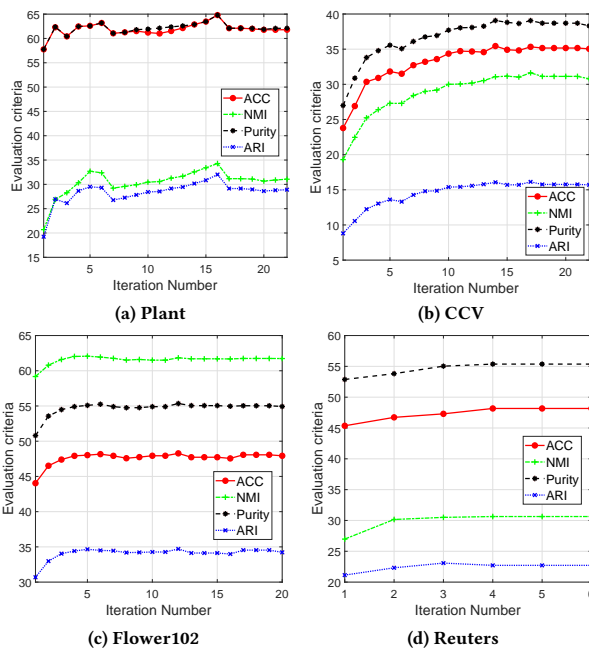
(d) Reuters

Figure 5: Evolution of clustering performance during the iterations on Plant, CCV, Flower102, and Reuters.

complexity requires more time on several datasets, which is mainly due to our more complex optimization, we believe that more computational cost is a worthwhile sacrifice for much better clustering performance.

3) Compared to other methods with $O(n^3)$ complexity, our model exhibits significant superiority in effectiveness and efficiency.

4.6 Comparison to Empirical HKMC Method

To further compare our MKC-DNM algorithm and the strongest competitor HKMC method. Figure 4 plots the results of ours and HKMC on BBCSport, CCV and Flower102.

Following the original hyper-parameter setting of [41], i.e. c_1 varies in [3k, 4k, $\cdots$, 10k] and c_2 varies in [2k, 3k, $\cdots$, 10k] with total 72 times computation by grid search. As the result shows,

our parameter-free MKC-DNM surpasses HKMC with an obvious margin in the wide searching region. Although the proposed MKC-DNM may exhibit slight weaker performance at several searching regions, we emphasize that our model does not require time-consuming hyper-parameter tuning but achieves promising performance, which directly serves for unsupervised clustering.

4.7 Evolution of Clustering Performance

To evaluate the effectiveness of our learning strategy to minimize dual noise within partition matrices in kernel space. Figure 5 plots the evolution of ACC, NMI, purity, and ARI on Plant, CCV, Flower102, and Reuters. As can be seen, four metrics increase and keep stable during the iterations, which sufficiently illustrates the effectiveness of our algorithm.

5 CONCLUSION

This work investigates an essential issue that how to minimize the noise inside the partition matrix. In this paper, we propose a novel parameter-free MKC algorithm with dual noise minimization to tackle this issue. Specifically, we first rigorously define the dual noise mathematically and separate it into dual parts, i.e. N-noise and C-noise, then we develop a unified and compact framework to minimize them. We design an efficient two-step iterative strategy to solve the model. To our best knowledge, it is the first time to investigate dual noise within the partition in the kernel space, distinguishing our work from existing researches. We observe that dual noise will pollute the block diagonal structures and incur the degeneration of clustering performance, and C-noise exhibits stronger destruction than N-noise. Minimizing dual noise significantly improves the clustering performance. Extensive experiments illustrate the effectiveness and efficiency of our proposed algorithm compared to the existing methods.

6 ACKNOWLEDGMENTS

REFERENCES

[1] Feiping Nie, Xiaoqian Wang, and Heng Huang. Clustering and projected clustering with adaptive neighbors. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 977–986, 2014.

[2] Yazhou Ren, Shudong Huang, Peng Zhao, Minghao Han, and Zenglin Xu. Self-paced and auto-weighted multi-view clustering. *Neurocomputing*, 383:248–256, 2020.

[3] Feiping Nie, Xiaoqian Wang, Michael Jordan, and Heng Huang. The constrained laplacian rank algorithm for graph-based clustering. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[4] Yue Liu, Xihong Yang, Sihang Zhou, and Xinwang Liu. Simple contrastive graph clustering. *arXiv preprint arXiv:2205.07865*, 2022.

[5] Yazhou Ren, Ni Wang, Mingxia Li, and Zenglin Xu. Deep density-based image clustering. *Knowledge-Based Systems*, 197:105841, 2020.

[6] Changqing Zhang, Qinghua Hu, Huazhu Fu, Pengfei Zhu, and Xiaochun Cao. Latent multi-view subspace clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4279–4287, 2017.

[7] Jie Xu, Yazhou Ren, Huayi Tang, Xiaorong Pu, Xiaofeng Zhu, Ming Zeng, and Lifang He. Multi-vae: Learning disentangled view-common and view-peculiar visual representations for multi-view clustering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9234–9243, 2021.

[8] Xiaojun Chang, Feiping Nie, Zhigang Ma, Yi Yang, and Xiaofang Zhou. A convex formulation for spectral shrunk clustering. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[9] Kun Zhan, Changqing Zhang, Junpeng Guan, and Junsheng Wang. Graph learning for multiview clustering. *IEEE transactions on cybernetics*, 48(10):2887–2895, 2017.

[10] Jie Wen, Yong Xu, and Hong Liu. Incomplete multiview spectral clustering with adaptive graph learning. *IEEE transactions on cybernetics*, 50(4):1418–1429, 2018.

[11] Kun Zhan, Chaoxi Niu, Changlu Chen, Feiping Nie, Changqing Zhang, and Yi Yang. Graph structure fusion for multiview clustering. *IEEE Transactions on Knowledge and Data Engineering*, 31(10):1984–1993, 2018.

[12] Zhao Kang, Haiqi Pan, Steven CH Hoi, and Zenglin Xu. Robust graph learning from noisy data. *IEEE transactions on cybernetics*, 50(5):1833–1843, 2019.

[13] Shudong Huang, Zhao Kang, and Zenglin Xu. Auto-weighted multi-view clustering via deep matrix decomposition. *Pattern Recognition*, 97:107015, 2020.

[14] Shudong Huang, Ivor W Tsang, Zenglin Xu, Jiancheng Lv, and Quanhui Liu. Cdd: Multi-view subspace clustering via cross-view diversity detection. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 2308–2316, 2021.

[15] Siwei Wang, Xinwang Liu, Xinzhong Zhu, Pei Zhang, Yi Zhang, Feng Gao, and En Zhu. Fast parameter-free multi-view subspace clustering with consensus anchor guidance. *IEEE Transactions on Image Processing*, 31:556–568, 2022.

[16] Siwei Wang, Xinwang Liu, Suyuan Liu, Jiaqi Jin, Wenxuan Tu, Xinzhong Zhu, and En Zhu. Align then fusion: Generalized large-scale multi-view clustering with anchor matching correspondences. *arXiv preprint arXiv:2205.15075*, 2022.

[17] Liping Jing, Michael K Ng, and Tieyong Zeng. Dictionary learning-based subspace structure identification in spectral clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 24(8):1188–1199, 2013.

[18] Feiping Nie, Guohao Cai, and Xuelong Li. Multi-view clustering and semi-supervised classification with adaptive neighbours. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[19] Liping Jing, Lixin Zhou, Michael K Ng, and J Zhexue Huang. Ontology-based distance measure for text clustering. In *Proc. of SIAM SDM workshop on text mining, Bethesda, Maryland, USA*, 2006.

[20] Suyuan Liu, Siwei Wang, Pei Zhang, Kai Xu, Xinwang Liu, Changwang Zhang, and Feng Gao. Efficient one-pass multi-view subspace clustering with consensus anchors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7576–7584, 2022.

[21] Changqing Zhang, Yajie Cui, Zongbo Han, Joey Tianyi Zhou, Huazhu Fu, and Qinghua Hu. Deep partial multi-view learning. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[22] Yue Liu, Sihang Zhou, Xinwang Liu, Wenxuan Tu, and Xihong Yang. Improved dual correlation reduction network. *arXiv preprint arXiv:2202.12533*, 2022.

[23] Jie Wen, Zheng Zhang, Yong Xu, Bob Zhang, Lunke Fei, and Guo-Sen Xie. Cdimc-net: Cognitive deep incomplete multi-view clustering network. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3230–3236, 2021.

[24] Siwei Wang, Xinwang Liu, Li Liu, Wenxuan Tu, Xinzhong Zhu, Jiyuan Liu, Sihang Zhou, and En Zhu. Highly-efficient incomplete large-scale multi-view clustering with consensus bipartite graph. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9776–9785, 2022.

[25] Bin Zhao, James T Kwok, and Changshui Zhang. Multiple kernel clustering. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 638–649. SIAM, 2009.

[26] Changshui Zhang, Feiping Nie, and Shiming Xiang. A general kernelization framework for learning algorithms based on kernel pca. *Neurocomputing*, 73(4-6):959–967, 2010.

[27] Zhao Kang, Chong Peng, and Qiang Cheng. Kernel-driven similarity learning. *Neurocomputing*, 267:210–219, 2017.

[28] Shudong Huang, Zhao Kang, Ivor W Tsang, and Zenglin Xu. Auto-weighted multi-view clustering via kernelized graph learning. *Pattern Recognition*, 88:174–184, 2019.

[29] Zhao Kang, Liangjian Wen, Wenyu Chen, and Zenglin Xu. Low-rank kernel learning for graph-based clustering. *Knowledge-Based Systems*, 163:510–517, 2019.

[30] Zhenwen Ren, Quansen Sun, and Dong Wei. Multiple kernel clustering with kernel k-means coupled graph tensor learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9411–9418, 2021.

[31] John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

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

[33] Hsin-Chien Huang, Yung-Yu Chuang, and Chu-Song Chen. Multiple kernel fuzzy clustering. *IEEE Transactions on Fuzzy Systems*, 20(1):120–134, 2012.

[34] Xinwang Liu, Yong Dou, Jianping Yin, Lei Wang, and En Zhu. Multiple kernel k-means clustering with matrix-induced regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[35] Zhao Kang, Chong Peng, and Qiang Cheng. Twin learning for similarity and clustering: a unified kernel approach. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 2080–2086, 2017.

[36] Xinwang Liu, Sihang Zhou, Yueqing Wang, Miaomiao Li, Yong Dou, En Zhu, and Jianping Yin. Optimal neighborhood kernel clustering with multiple kernels. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[37] Zhao Kang, Yiwei Lu, Yuanzhang Su, Changsheng Li, and Zenglin Xu. Similarity learning via kernel preserving embedding. In *proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4057–4064, 2019.

[38] Liang Li, Siwei Wang, Xinwang Liu, En Zhu, Li Shen, Kenli Li, and Keqin Li. Local sample-weighted multiple kernel clustering with consensus discriminative graph. *arXiv preprint arXiv:2207.02846*, 2022.

[39] Siwei Wang, Xinwang Liu, En Zhu, Chang Tang, Jiyuan Liu, Jingtao Hu, Jingyuan Xia, and Jianping Yin. Multi-view clustering via late fusion alignment maximization. In *The Twenty-eighth International Joint Conference on Artificial Intelligence*, pages 3778–3784, 2019.

[40] Xinwang Liu, Xinzhong Zhu, Miaomiao Li, Lei Wang, Chang Tang, Jianping Yin, Dinggang Shen, Huaimin Wang, and Wen Gao. Late fusion incomplete multi-view clustering. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[41] Jiyuan Liu, Xinwang Liu, Yuexiang Yang, Siwei Wang, and Sihang Zhou. Hierarchical multiple kernel clustering. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI*, pages 2–9, 2021.

[42] Xinwang Liu, Li Liu, Qing Liao, Siwei Wang, Yi Zhang, Wenxuan Tu, Chang Tang, Jiyuan Liu, and En Zhu. One pass late fusion multi-view clustering. In *International Conference on Machine Learning*, pages 6850–6859. PMLR, 2021.

[43] Siwei Wang, Xinwang Liu, Li Liu, Sihang Zhou, and En Zhu. Late fusion multiple kernel clustering with proxy graph refinement. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[44] Tiejian Zhang, Xinwang Liu, Lei Gong, Siwei Wang, Xin Niu, and Li Shen. Late fusion multiple kernel clustering with local kernel alignment maximization. *IEEE Transactions on Multimedia*, pages 1–1, 2021.

[45] Liping Jing, Michael K Ng, and Joshua Zhexue Huang. An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data. *IEEE Transactions on knowledge and data engineering*, 19(8):1026–1041, 2007.

[46] Feiping Nie, Zinan Zeng, Ivor W Tsang, Dong Xu, and Changshui Zhang. Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering. *IEEE Transactions on Neural Networks*, 22(11):1796–1808, 2011.

[47] Jie Wen, Ke Yan, Zheng Zhang, Yong Xu, Junqian Wang, Lunke Fei, and Bob Zhang. Adaptive graph completion based incomplete multi-view clustering. *IEEE Transactions on Multimedia*, 23:2493–2504, 2020.

[48] Changqing Zhang, Huazhu Fu, Qinghua Hu, Xiaochun Cao, Yuan Xie, Dacheng Tao, and Dong Xu. Generalized latent multi-view subspace clustering. *IEEE transactions on pattern analysis and machine intelligence*, 42(1):86–99, 2018.

[49] Grigorios Tzortzis and Aristidis Likas. The global kernel k-means clustering algorithm. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1977–1984. IEEE, 2008.

[50] Werner H Greub. *Linear algebra*, volume 23. Springer Science & Business Media, 2012.

[51] Igor Griva, Stephen G Nash, and Ariela Sofer. *Linear and nonlinear optimization*, volume 108. Siam, 2009.

[52] Kazuo Murota. *Discrete convex analysis*, volume 10 of *SIAM monographs on discrete mathematics and applications*. SIAM, 2003.

[53] Shmuel Onn. Convex discrete optimization. In Vasek Chvátal, editor, *Combinatorial Optimization - Methods and Applications*, volume 31 of *NATO Science for Peace and Security Series - D: Information and Communication Security*, pages 183–228. IOS Press, 2011.

[54] Zhao Kang, Wangtao Zhou, Zhitong Zhao, Junming Shao, Meng Han, and Zenglin Xu. Large-scale multi-view subspace clustering in linear time. In *Proceedings of*

the *AAAI conference on Artificial Intelligence*, volume 34, pages 4412–4419, 2020.

[55] James C. Bezdek and Richard J. Hathaway. Convergence of alternating optimization. *Neural, Parallel Sci. Comput.*, 11(4):351–368, 2003.

[56] Feiping Nie, Jing Li, Xuelong Li, et al. Self-weighted multiview clustering with multiple graphs. In *The Twenty-sixth International Joint Conference on Artificial Intelligence*, pages 2564–2570, 2017.

[57] Zhenwen Ren and Quansen Sun. Simultaneous global and local graph structure preserving for multiple kernel clustering. *IEEE transactions on neural networks and learning systems*, 32(5):1839–1851, 2020.

[58] Zhenwen Ren, Simon X Yang, Quansen Sun, and Tao Wang. Consensus affinity graph learning for multiple kernel clustering. *IEEE Transactions on Cybernetics*, 2020.

[59] Shudong Huang, Yazhou Ren, and Zenglin Xu. Robust multi-view data clustering with multi-view capped-norm k-means. *Neurocomputing*, 311:197–208, 2018.

[60] Jie Wen, Zheng Zhang, Yong Xu, Bob Zhang, Lunke Fei, and Hong Liu. Unified embedding alignment with missing views inferring for incomplete multi-view clustering. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5393–5400, 2019.

[61] Yazhou Ren, Uday Kamath, Carlotta Domeniconi, and Zenglin Xu. Parallel boosted clustering. *Neurocomputing*, 351:87–100, 2019.

[62] Jie Wen, Zheng Zhang, Zhao Zhang, Lunke Fei, and Meng Wang. Generalized incomplete multiview clustering with flexible locality structure diffusion. *IEEE transactions on cybernetics*, 51(1):101–114, 2020.

[63] Changqing Zhang, Huazhu Fu, Si Liu, Guangcan Liu, and Xiaochun Cao. Low-rank tensor constrained multiview subspace clustering. In *Proceedings of the IEEE international conference on computer vision*, pages 1582–1590, 2015.

A PROOF OF LEMMA

Recall the definition of dual noise, i.e.

$$v\left(\mathbf{E}_p^{\mathrm{N}}\right) \in N\left(\mathbf{H}^{*\top}\right) = \left\{\mathbf{x} \mid \mathbf{H}^{*\top}\mathbf{x} = \mathbf{0}\right\}, \quad \forall p \in \Delta_m, \tag{24}$$

$$v\left(\mathbf{E}_p^{\mathrm{C}}\right) \in C\left(\mathbf{H}^*\right) = \left\{\mathbf{y} \mid \exists\, \mathbf{x},\ \text{s.t.}\ \mathbf{y} = \mathbf{H}^*\mathbf{x}\right\}, \quad \forall p \in \Delta_m, \tag{25}$$

where $\Delta_m = \{1, 2, \cdots, m\}$.

A.1 Proof of Lemma 1

LEMMA A.1. $\mathrm{Tr}\left(\mathbf{E}_p^{\mathrm{N}}\mathbf{H}^*\mathbf{H}^{*\top}\right) = 0, \quad \forall p \in \Delta_m.$

PROOF. The spectral decomposition of $\mathbf{E}_p^{\mathrm{N}}$ is formulated as

$$\mathbf{E}_p^{\mathrm{N}} = \sum_i \lambda_i v_i v_i^{\top}. \tag{26}$$

where v_i is the unit eigenvector with corresponding eigenvalue λ_i.

According to Eq. (24), we have $\lambda_i \mathbf{H}^{*\top} v_i = 0$. We can further derive that:

$$\mathrm{Tr}\left(\mathbf{E}_p^{\mathrm{N}}\mathbf{H}^*\mathbf{H}^{*\top}\right) = \mathrm{Tr}\left(\sum_i \lambda_i v_i v_i^{\top} \mathbf{H}^*\mathbf{H}^{*\top}\right)$$
$$= \mathrm{Tr}\left(\sum_i v_i^{\top}\mathbf{H}^*\left(\lambda_i \mathbf{H}^{*\top} v_i\right)\right) = 0. \tag{27}$$

This completes the proof. □

A.2 Proof of Lemma 2

LEMMA A.2. $\mathrm{Tr}\left(\mathbf{E}_p^{\mathrm{C}}\mathbf{H}^*\mathbf{H}^{*\top}\right) = \mathrm{Tr}\left(\mathbf{E}_p^{\mathrm{C}}\right), \quad \forall p \in \Delta_m.$

PROOF. The spectral decomposition of $\mathbf{E}_p^{\mathrm{C}}$ is formulated as

$$\mathbf{E}_p^{\mathrm{C}} = \sum_i \lambda_i v_i v_i^{\top}. \tag{28}$$

where v_i is the unit eigenvector with corresponding eigenvalue λ_i.

According to Eq. (25), we have $v_i = \mathbf{H}^* x_i$, where x_i is a unit vector. We can further derive that:

$$\mathrm{Tr}\left(\mathbf{E}_p^{\mathrm{C}}\mathbf{H}^*\mathbf{H}^{*\top}\right) = \mathrm{Tr}\left(\sum_i \lambda_i v_i v_i^{\top}\mathbf{H}^*\mathbf{H}^{*\top}\right)$$
$$= \mathrm{Tr}\left(\sum_i \lambda_i \left(\mathbf{H}^{*\top} v_i\right)^{\top}\left(\mathbf{H}^{*\top} v_i\right)\right) \tag{29}$$
$$= \mathrm{Tr}\left(\sum_i \lambda_i x_i^{\top} x_i\right) = \sum_i \lambda_i = \mathrm{Tr}\left(\mathbf{E}_p^{\mathrm{C}}\right).$$

This completes the proof. □

A.3 Proof of Lemma 3

LEMMA A.3. $\mathbf{E}_p^{\mathrm{N}}$ is positive semi-definite (PSD) and $\mathbf{E}_p^{\mathrm{C}}$ is negative semi-definite (NSD).

PROOF. For all $\mathbf{x} \in \mathbb{R}^n$, we separate it into two parts:

$$\mathbf{x} = \mathbf{x}_{\mathrm{N}} + \mathbf{x}_{\mathrm{C}}, \tag{30}$$

where $\mathbf{x}_{\mathrm{N}} \in N\left(\mathbf{H}^{*\top}\right)$ and $\mathbf{x}_{\mathrm{C}} \in C\left(\mathbf{H}^*\right)$ are existing and unique. According to Eq. (25), we have $\mathbf{x}_{\mathrm{C}} = \mathbf{H}^*\mathbf{y}$, where $\|\mathbf{y}\|_2^2 = \|\mathbf{x}_{\mathrm{C}}\|_2^2$. Then we have

$$\mathbf{H}^{*\top}\mathbf{x}_{\mathrm{N}} = \mathbf{0},\ \mathbf{E}_p^{\mathrm{C}}\mathbf{x}_{\mathrm{N}} = \mathbf{0},\ \mathbf{E}_p^{\mathrm{N}}\mathbf{x}_{\mathrm{C}} = \mathbf{0}. \tag{31}$$

We can further derive that:

$$\mathbf{x}^{\top}\mathbf{E}_p^{\mathrm{N}}\mathbf{x} = \mathbf{x}_{\mathrm{N}}^{\top}\mathbf{E}_p^{\mathrm{N}}\mathbf{x}_{\mathrm{N}}$$
$$= \mathbf{x}_{\mathrm{N}}^{\top}\mathbf{U}_p\mathbf{U}_p^{\top}\mathbf{x}_{\mathrm{N}} - \mathbf{x}_{\mathrm{N}}^{\top}\left(\mathbf{H}^*\mathbf{H}^{*\top} + \mathbf{E}_p^{\mathrm{C}}\right)\mathbf{x}_{\mathrm{N}} \tag{32}$$
$$= \mathbf{x}_{\mathrm{N}}^{\top}\mathbf{U}_p\mathbf{U}_p^{\top}\mathbf{x}_{\mathrm{N}} \geq 0,$$

and

$$\mathbf{x}^{\top}\mathbf{E}_p^{\mathrm{C}}\mathbf{x} = \mathbf{x}_{\mathrm{C}}^{\top}\mathbf{E}_p^{\mathrm{C}}\mathbf{x}_{\mathrm{C}}$$
$$= \mathbf{x}_{\mathrm{C}}^{\top}\mathbf{U}_p\mathbf{U}_p^{\top}\mathbf{x}_{\mathrm{C}} - \mathbf{x}_{\mathrm{C}}^{\top}\left(\mathbf{H}^*\mathbf{H}^{*\top} + \mathbf{E}_p^{N}\right)\mathbf{x}_{\mathrm{C}}$$
$$= \mathbf{x}_{\mathrm{C}}^{\top}\mathbf{U}_p\mathbf{U}_p^{\top}\mathbf{x}_{\mathrm{C}} - \mathbf{x}_{\mathrm{C}}^{\top}\mathbf{H}^*\mathbf{H}^{*\top}\mathbf{x}_{\mathrm{C}} \tag{33}$$
$$= \mathbf{x}_{\mathrm{C}}^{\top}\mathbf{U}_p\mathbf{U}_p^{\top}\mathbf{x}_{\mathrm{C}} - \mathbf{y}^{\top}\mathbf{y}$$
$$\leq \mathbf{x}_{\mathrm{C}}^{\top}\mathbf{x}_{\mathrm{C}} - \mathbf{y}^{\top}\mathbf{y} = 0.$$

This completes the proof. □