An Integer Programming Approach To Subspace Clustering With Missing Data

Akhilesh Soni, Jeff Linderoth, James Luedtke
Department of Industrial and Systems Engineering, University of Wisconsin-Madison, akhileshsoni95@gmail.com, linderoth@wisc.edu,jim.luedtke@wisc.edu

Daniel Pimentel-Alarcón
Department of Biostatistics and Medical Informatics, University of Wisconsin-Madison , pimentelalar@wisc.edu

In the *Subspace Clustering with Missing Data* (SCMD) problem, we are given a collection of n partially observed d-dimensional vectors. The data points are assumed to be concentrated near a union of low-dimensional subspaces. The goal of SCMD is to cluster the vectors according to their subspace membership and recover the underlying basis, which can then be used to infer their missing entries. State-of-the-art algorithms for SCMD can fail on instances with a high proportion of missing data, full-rank data, or if the underlying subspaces are similar to each other. We propose a novel integer programming approach for SCMD. The approach is based on dynamically determining a set of candidate subspaces and optimally assigning points to selected subspaces. The problem structure is identical to the classical facility-location problem, with subspaces playing the role of facilities and data points that of customers. We propose a column-generation approach for identifying candidate subspaces combined with a Benders decomposition approach for solving the linear programming relaxation of the formulation. An empirical study demonstrates that the proposed approach can achieve better clustering accuracy than state-of-the-art methods when the data is high-rank, the percentage of missing data is high, or the subspaces are similar.

Key words: subspace clustering; matrix completion; integer programming

1. Introduction

We consider a real-valued matrix $X \in \mathbb{R}^{d\times n}$ in which each column $X_1, X_2, \ldots, X_n$ is assumed to lie near one of K unknown subspaces, $\mathcal{S}_i$ with dimension $r_i < d$, $i = 1, \ldots, K$. Given a set of Ω of observed entries of the matrix X, *subspace clustering with missing data* (SCMD) is the task of identifying clusters of the columns of X_Ω belonging to the same subspace and inferring the subspace associated with each cluster of columns.

If the clustering of columns is known, then the subspace associated with each cluster can be estimated by applying methods for low-rank matrix completion (LRMC) on the columns in each cluster (Candès and Recht 2009, Candes and Tao 2010, Cai et al. 2010, Balzano et al. 2010, Recht 2011, Pimentel-Alarcón et al. 2015, Nguyen et al. 2019). The SCMD problem has applications in machine learning for image classification (Lecun et al. 1998, Zapata et al. 2007), motion segmen-

tation (Tron and Vidal 2007, Rao et al. 2010), and recommendation systems (Ramlatchan et al. 2018).

In this work, we propose a novel mixed-integer linear programming (MILP) solution framework for the SCMD problem that is based on dynamically determining a set of candidate subspaces and optimally assigning columns (also called data points) to the closest selected subspace. We refer to our method as MISS-DSG : Mixed Integer Subspace Selector with Dynamic Subspace Generation. A key contribution of our approach is a method for identifying, in a rigorous manner, a suitable set of candidate subspaces to include in the formulation. We cast this subspace generation problem as a nonlinear, nonconvex optimization problem and propose a gradient-based approximate solution approach. We also use Benders decomposition to solve the linear programming (LP) relaxation of the MILP, which allows our framework to accommodate both a large number of candidate subspaces and a large number of data points. The model has the advantage of integrating the subspace generation and clustering in a single, unified optimization framework without requiring any hyperparameter tuning when the number of subspaces and subspaces dimensions are known. When the number and/or dimensions of the subspaces are unknown, we propose add a regularization term to the objective based on the effective dimension of the selected subspaces. Our computational study demonstrates that the proposed method can achieve higher clustering accuracy than state-of-the-art methods when the underlying matrix X is high-rank, the percentage of missing data is high, or the subspaces are similar to each other.

Prior Work. Abdolali and Gillis (2021) provide a a review of subspace clustering methods in the case that all entries of X are observed. Most of the methods for subspace clustering with missing data have been adapted from the methods initially proposed for fully-observed data. The tightest known conditions for union of subspaces identifiability with missing data have been established by Pimentel-Alarcón and Nowak (2016), where the authors show that for ambient dimension d and low dimensional subspaces of dimension r, observing $O(rd)$ columns per subspace is both necessary and sufficient for subspace identification.

The dominant approach in subspace clustering is based on a self-expressiveness property, originally proposed for fully-observed data by Elhamifar and Vidal (2009). Self-expressive methods learn a sparse representation of the data by solving an optimization problem of the form:

$$C^* = \arg\min \|X - XC\|_F^2 + \lambda\rho(C) \qquad \text{s.t. } \operatorname{diag}(C) = 0. \tag{1}$$

Self-expressive methods have been studied extensively for different choices of $\rho(\cdot)$, e.g., ℓ_1, ℓ_2, and nuclear norm (Elhamifar and Vidal 2009, Liu et al. 2010, Lu et al. 2012, Zhuang et al. 2012, Lu et al. 2013, Elhamifar and Vidal 2013, Panagakis and Kotropoulos 2014, You et al. 2016, Wang

et al. 2019). The value C^*_{ij} can be interpreted as a link or connection between data points i and j. The segmentation is obtained by applying spectral clustering on a graph G with adjacency matrix $A = |C^*| + |C^*|^T$, which uses k-means cluster of the eigenvectors of the Laplacian of G (Ng et al. 2001). Self-expressive methods for subspace clustering have been extended to the case of missing data. Let Ω_j denote the set of observed components of vector $j \in [n]$ and $I_\Omega \in \{0,1\}^{d\times n}$ be the indicator matrix of observed entries such that $[I_\Omega]_{ij} = 1$ if $(i,j) \in \Omega$, and 0 otherwise. Let $\circ$ denote the Hadamard product. Yang et al. (2015) proposed to zero-fill the missing entries in X to get X_{ZF} and to solve (1) while restricting the loss to observed entries, i.e.,

$$X_{ZF} = \begin{cases} X_{ij}, & \text{if } (i,j) \in \Omega, \text{i.e., } X_{ij} \text{ is observed} \\ 0, & \text{if } (i,j) \notin \Omega, \text{i.e., } X_{ij} \text{ is not observed} \end{cases} \tag{2}$$

$$C^* = \arg\min \|I_\Omega \circ (X_{ZF} - X_{ZF}C)\|_F^2 + \lambda\rho(C) \qquad \text{s.t. } \operatorname{diag}(C) = 0. \tag{3}$$

Tsakiris and Vidal (2018) and Charles et al. (2018) studied the theoretical conditions under which the solution to (3) is subspace-preserving, i.e., each data point is only connected to points lying in the same subspace.

Representative of a class of methods that alternate between subspace estimation and assignment, Yang et al. (2015) proposed to apply a matrix completion algorithm to recover the missing entries in X and then solve (1). This approach has been observed to fail when the data matrix is high-rank, i.e., $\sum_{i=1}^{K} r_i \approx d$. Lane et al. (2019) proposed to alternate between subspace clustering and group wise low-rank matrix completion (gLRMC). Balzano et al. (2012) use GROUSE (Balzano et al. 2010) for subspace estimation and assign each point to the orthogonally closest subspace. They use probabilistic farther insertion for initializing K subspaces. In all alternating methods, the subspace estimation process is often faulty when an estimated cluster has points from different subspaces. In an extensive empirical evaluation of existing SCMD algorithms, Lane et al. (2019) concluded that the zero-filled elastic net subspace clustering method (You et al. 2016) when alternated with low-rank matrix completion showed the overall best performance. This method is referred to as Alt-PZF-EnSC+gLRMC.

Some methods pose the two problems of subspace estimation and assignment in a joint optimization framework, often resulting in complex, nonconvex problems (Li and Vidal 2016, Elhamifar 2016, Fan and Chow 2017). Matrix factorization approaches have also been adopted to SCMD (Pimentel et al. 2014, Pimentel-Alarcón et al. 2016). Empirical experiments in (Lane et al. 2019) showed that these methods are outperformed by the alternating methods in terms of clustering error.

The work most closely related to our approach are the MILP-based methods for subspace clustering with fully-observed data. Lazic et al. (2009) were the first to propose an MILP-based method

for subspace clustering called Facility Location for Subspace Segmentation (FLoSS). FLoSS generates the candidate subspaces at random and formulates the subspace clustering problem as an MILP. The goal of the MILP is to minimize the orthogonal distances of data points to candidate subspaces such that K subspaces are selected, and each vector is assigned to a selected subspace. Lee and Cheong (2013) extended the FLoSS model to Minimal Basis FLoSS (MB-FLoSS), where the subspace hypothesis generation strategy is based on finding the minimal basis subspace representation for the data matrix and relies on Low Rank Representation (LRR)(Liu et al. 2010). Hu et al. (2015) proposed the concept of constrained subspace model. They integrated the facility-based model with manifold and spatial regularity constraints to develop a constrained subspace modeling framework. The number of candidate subspaces is small (≤ 50) in their experiments. The method becomes inefficient when the number of candidate subspaces is higher, and the approach heavily relies on the efficiency of initial candidate subspaces generated for which they use over segmentation in LRR (Liu et al. 2010). In particular, they generate more subspaces than the ground truth (e.g, $2 \times K$) with LRR, and then use MILP to select K of them. None of these existing MILP-based approaches account for missing data or scale to instances with a large number of candidate subspaces. Moreover, all of the approaches require that candidate subspaces are explicitly enumerated as an input to the model, and either rely on random sampling or other subspace clustering algorithms for generating candidate subspaces. Hence, these methods are incapable of correcting themselves based on the clustering quality. Our approach MISS-DSG can handle a large set of candidate subspaces through the use of Benders decomposition and identifies new candidate subspaces dynamically through the use of column generation. Casting SCMD as an MILP offers several other advantages. The formulation can easily be extended to incorporate prior information about the data, such as vectors lying in the same or different subspaces and bounds on number of subspaces.

The paper begins with a description of the MILP formulation in Section 2. Section 3 discusses our decomposition approach to solve the model, and Section 5 presents experimental results that show the effectiveness of our framework.

2. MILP Formulations

We consider a real-valued matrix $X \in \mathbb{R}^{d \times n}$ whose columns are concentrated near a union of K subspaces with dimensions $r_1, r_2, \ldots, r_K$. For an integer T, we denote $[T] := \{1, 2, \ldots, T\}$. We denote data vector (column) $j \in [n]$ as X_j. We assume that we observe a subset $\Omega \subseteq [d] \times [n]$ of the entries of X, and given this data the goal of subspace clustering with missing data (SCMD) is to identify the K subspaces together with assignment of data points to subspaces. This consequently leads to a clustering of points and a method for estimating missing entries of X. In Section 2.1, we assume

that subspaces dimension $r_1, \ldots, r_K$ are known. We relax this assumption in Section 2.2 and let the model self-determine the subspaces dimensions with the help of a regularized objective. The matrix X is referred to as *low-rank* when $\sum_{k=1}^{K} r_k \ll \min\{d, n\}$ and as *high-rank* when $\sum_{k=1}^{K} r_k \approx \min\{d, n\}$.

Our approach is based on iteratively building a (potentially very-large) collection of candidate subspaces. MILP is employed to simultaneously select the best set of K candidate subspaces and assign each column of X to its closest selected subspace. For each candidate subspace $t \in [T]$, we let $U_t \in \mathbb{R}^{d \times r_t}$ be a basis for its column subspace, where r_t is the dimension of candidate subspace t. We define the distance of vector $j \in [n]$ to a candidate subspace $t \in [T]$ as the sum of the squared residuals on the observed entries:

$$\Delta_{jt} := \min_{v \in \mathbb{R}^r} \Big\{ \sum_{i:(i,j)\in\Omega} (X_{ij} - (U_t v)_i)^2 \Big\}. \tag{4}$$

These quantities have a closed-form solution in terms of a simple projection operator (Balzano et al. 2012). In particular, let $U_{\Omega,j}$ denote the restriction of the subspace U to the rows observed in column j, and define the projection operator $P_{U_{\Omega,j}} := U_{\Omega,j}(U_{\Omega,j}^T U_{\Omega,j})^{-1} U_{\Omega,j}^T$. Then the squared residual Δ_{jt} can be obtained as

$$\Delta_{jt} = \|X_{\Omega,j} - P_{(U_t)_{\Omega,j}}(X_{\Omega,j})\|_2^2. \tag{5}$$

For fully-observed data, this is a natural choice for cost function since its value is zero if vector j is in subspace t. However, with missing data, the choice of cost function becomes less clear since zero residual on observed entries for Δ_{jt} does not necessarily imply that vector j lies perfectly on subspace t. Balzano et al. (2012) showed that for a given fully observed vector $X_j \in \mathbb{R}^d$, if

$$\|X_j - P_{U_0}(X_j)\| < \|X_j - P_{U_t}(X_j)\| \ \forall t \in [T]\backslash\{0\}, \tag{6}$$

then, with high probability, for the same data vector X_j but now partially observed on Ω (sampled uniformly from $[d]$ with replacement), if "enough" elements of the data point X_j are observed (see Balzano et al. (2012) for details on quantification of "enough"), then

$$\|X_{\Omega,j} - P_{(U_0)_{\Omega,j}}(X_{\Omega,j})\| < \|X_{\Omega,j} - P_{(U_t)_{\Omega,j}}(X_{\Omega,j})\| \ \forall t \in [T]\backslash\{0\}. \tag{7}$$

This implies that, with high probability, subspace assignment based on (6) is the same as the one based on (7). We refer reader to Balzano et al. (2012) for more details. We also note that a different cost model could also be incorporated into our framework.

Next, we describe a model based on selecting K subspaces from a given collection of $[T]$ subspaces for both known and unknown subspaces dimensions in Section 2.1 and 2.2 respectively. In Section 3.1, we discuss how to solve the proposed model for a fixed set of subspaces using Benders

decomposition. This allows to solve the model efficiently for large n and T. In Section 3.2, we discuss how to generate new candidate subspaces dynamically with a column generation approach. We finally discuss our unified framework MISS-DSG in Section 3.3.

Let $x_{jt} \in \{0,1\}, \forall j \in [n], t \in [T]$ be a binary assignment variable that takes value 1 if vector j is assigned to subspace t, and $z_t \in \{0,1\}, \forall t \in [T]$ be a binary selection variable with $z_t = 1$ if subspace t is selected. The assignment of points to selected subspaces is similar to the facility location problem, where the goal is to select which facilities to open and to assign each customer to an open facility. In our SCMD formulation, subspaces play the role of facilities, and vectors play the role of customers. Previously proposed MILP methods in the literature have this same facility-location structure (Lazic et al. 2009, Lee and Cheong 2013, Hu et al. 2015). However, our model has some key differences:

(a) We allow missing data while existing MILP approaches restrict to fully observed data.
(b) Our framework generates subspaces dynamically while existing approaches are heavily dependent on initially generating candidate subsspaces.
(c) Our framework is capable of handling a larger number of candidate subspaces than existing methods through the use of Benders decomposition.

For improved readability, in Section 2.1 we first discuss the formulation for the simpler case where subspaces dimensions are assumed to be known and equal, i.e., $r_1 = r_2 = \cdots = r_k = r$. We then relax this assumption in Section 2.2.

2.1. Known subspaces dimension

Given T candidate subspaces each of dimension r, we formulate the SCMD problem as the MILP (Lazic et al. 2009, Hu et al. 2015):

$$\min_{x \in \{0,1\}^{n \times T}, z \in \{0,1\}^T} \sum_{t \in [T]} \sum_{j \in [n]} \Delta_{jt} x_{jt} \tag{8a, MILP}$$

$$\sum_{t \in [T]} x_{jt} = 1, \quad \forall j \in [n] \tag{8b}$$

$$x_{jt} \leq z_t, \quad \forall j \in [n], t \in [T] \tag{8c}$$

$$\sum_{t \in [T]} z_t = K. \tag{8d}$$

The objective (8a) seeks the least cost assignment of vectors to subspaces. Constraints (8b) ensures that each vector is assigned to exactly one subspace, and constraints (8c) enforce that a vector can only be assigned to a selected subspace. Constraint (8d) requires that exactly K subspaces are selected. The major difference in (8) compared to the models proposed in (Lazic et al. 2009, Hu et al. 2015) is that the distance metric Δ_{jt} in (8) is based on partial assignment cost (5), as we do not assume data is fully observed.

2.2. Unknown Subspaces Dimension

A common occurrence in SCMD problems is that the number of subspaces and their dimension are unknown. The objective in model (8) may be in appropriate in this case, because it is likely to favor subspaces of higher dimension, since these are inherently more flexible and hence will lead to lower residuals. Thus, when the subspace dimensions are unknown, we propose to augment the objective in (8) with a complexity measure of the candidate subspaces, the effective dimension (ED). Huang et al. (2004) defined effective dimension for X on a union of subspace models $S = \cup_{k=1}^{K} S_k$ as follows:

$$ED(X, S) := \frac{1}{n} \sum_{k=1}^{K} r_k(d - r_k) + \frac{1}{n} \sum_{k=1}^{K} n_k r_k. \tag{9}$$

The first term in the definition of ED in (9), $r_k(d - r_k)$ is the complexity of U_t—the number of real numbers needed to specify a k dimensional subspace S_k in $\mathbb{R}^d$. The second term of (9), $n_k r_k$ is the number of real numbers needed to specify the r_k coordinates of the n_k sample points in the subspace S_k.

To allow for the model to trade-off accuracy with complexity, we add the ED term with a weight parameter λ in our objective as follows:

$$\min_{x \in \{0,1\}^{n \times T}, z \in \{0,1\}^T} \sum_{t \in [T]} \sum_{j \in [n]} \Delta_{jt} x_{jt} + \frac{\lambda}{n} \sum_{t \in T} \left(r_t(d - r_t) z_t + \sum_{j \in [n]} r_t x_{jt} \right) \tag{10a}$$

$$\sum_{t \in [T]} x_{jt} = 1, \qquad \forall j \in [n] \tag{10b}$$

$$x_{jt} \leq z_t, \qquad \forall j \in [n], t \in [T] \tag{10c}$$

$$\sum_{t \in [T]} z_t = K. \tag{10d}$$

Formulation (10) can handle subspaces of multiple dimensions and choose the best union of subspaces model by self-determining dimensions of subspaces. Constraint (10d) is included but can be removed if K is unknown. An important consideration in model (10) is the choice of regularization parameter λ which accounts for the trade-off between lower assignment cost and complexity of the selected subspaces. A smaller value of λ would promote model (10) to select higher complexity subspaces (basis with higher dimensions r_t) while a larger value of λ would promote the model to select lower complexity subspaces (basis with lower dimensions r_t). We discuss this in more detail in Section 5.5. Note that (8) is a special case of (10) with $\lambda = 0$.

3. Decomposition Algorithm

We next discuss how to solve formulation (10) for a given set of candidate subpsaces via Benders decomposition and how to dynamically generate a set of candidate subspaces to use in the formulation.

As is standard for solving a MILP, the first step in solving formulation (10) is to solve its LP relaxation. The LP relaxation of (10) is the problem created by replacing the integrality conditions $z_t, x_{jt} \in \{0,1\}$ with simple bound constraints $z_t, x_{jt} \in [0,1]$. The optimal solution value of the LP relaxation provides a lower bound on the optimal solution to (10). The number of candidate subspaces T and the number of points n may be quite large, so solving the LP relaxation could be a computational challenge. We discuss in Section 3.1 how Benders decomposition can address this challenge for solving the LP relaxation of (10). In Section 3.2, we describe our approach for dynamically generating additional candidate subspaces to include in the formulation, which is also done for the LP relaxation. Finally, in Section 3.3 we describe how these components are integrated into an overall approach for yielding feasible solutions to (10).

3.1. Benders Decomposition

Benders decomposition is a well-known technique to solve large LP problems that have special structure (Benders 1962). It has been applied to large-scale facility locations by Fischetti et al. (2017). Since our SCMD formulation (10) has the same structure, we can apply the same approach for solving its LP relaxation. The first step in the decomposition approach is to write a reformulation that eliminates the x_{jt} variables and adds a continuous variable w_j for each $j \in [n]$ that represents the assignment cost for vector j. The resulting reformulation of the LP relaxation of (10) is

$$\min_{w \in R^n, z \in [0,1]^T} \sum_{j \in [n]} w_j + \frac{\lambda}{n} \sum_{t \in T} r_t (d - r_t) z_t \tag{11a}$$

$$w_j \geq \Phi_j(z), \qquad \forall j \in [n] \tag{11b}$$

$$\sum_{t \in [T]} z_t = K, \tag{11c}$$

where $\Phi_j(\cdot)$ is defined as

$$\Phi_j(\hat{z}) = \min_x \Big\{ \sum_{t \in [T]} (\Delta_{jt} + \frac{\lambda}{n} r_t) x_t : \sum_{t \in [T]} x_t = 1, 0 \leq x_t \leq \hat{z}_t, \forall t \in [T] \Big\}. \tag{12}$$

The function $\Phi_j(\hat{z})$ computes the minimum (fractional) assignment cost for the vector $j \in [n]$ for a given (possibly fractional) vector of facility opening decisions $\hat{z} \in [0,1]^T$. The function $\Phi_j(\cdot)$ is piecewise-linear and convex, and Benders decomposition works by dynamically building a lower-bound approximation to $\Phi_j(\cdot)$ via the addition of Benders cuts.

To simplify notation in the derivation of the Benders cuts, we define $c_{jt} = \Delta_{jt} + \frac{\lambda}{n} r_t$ for $j \in [n]$ and $t \in [T]$. The optimization problem (12) used to evaluate $\Phi_j(\hat{z})$ has a closed-form solution. Moreover, its evaluation also gives sufficient information to derive the Benders cuts that define the lower-bounding approximation. For each $j \in [n]$, let $\{\sigma_1^j, \ldots, \sigma_T^j\}$ be a permutation of $[T]$ satisfying

$c_{j\sigma_1^j} \leq c_{j\sigma_2^j} \leq \cdots \leq c_{j\sigma_T^j}$, and let $t_j^* := \min\{t : \sum_{s=1}^{t} \hat{z}_{\sigma_s^j} \geq 1\}$ be the *critical index*. In other words, the *critical index* is the index of the costliest subspace to which any portion of vector j is assigned. As described in Fischetti et al. (2017), the Benders cut that can be used to lower-approximate the function $\Phi_j(\cdot)$ is

$$w_j + \sum_{i=1}^{t_j^*-1} (c_{j\sigma_{t_j^*}^j} - c_{j\sigma_i^j}) z_{\sigma_i^j} \geq c_{j\sigma_{t_j^*}^j}. \tag{13}$$

These inequalities are accumulated iteratively. Let p_j denote the number of Benders cuts included in the model at the current stage in the algorithm for each $j \in [n]$. Let t_{ji}^* denote the critical index for vector $j \in [n]$ associated with Benders cut $i \in [p_j]$, and let $c_{ji}^* = c_{j\sigma_{t_{ji}^*}^j}$ denote the critical cost for the j^{th} vector in cut $i \in [p_j]$. The Benders *master problem* is then

$$\min_{w,z} \sum_{j\in[n]} w_j + \frac{\lambda}{n} \sum_{t\in[T]} r_t(d - r_t) z_t \tag{14a}$$

$$w_j + \sum_{\ell=1}^{t_{ji}^*-1} (c_{ji}^* - c_{j\sigma_\ell^i}) z_{\sigma_\ell^j} \geq c_{ji}^*, \; \forall j \in [n], i \in [p_j], \tag{14b, α_{ji}}$$

$$\sum_{t\in[T]} z_t = K, \tag{14c, β}$$

$$0 \leq z_t \leq 1, \qquad \forall t \in [T]. \tag{14d, μ_t}$$

Here α, β and μ are dual variables corresponding to the respective constraints, and will play an important role in the column generation process described in Section 3.2. For $T > K$, LP (14) is feasible and bounded, and hence an optimal solution $(\hat{w}, \hat{z})$ exists. The subproblem (12) is solved to evaluate $\Phi_j(\hat{z})$ for each $j \in [n]$, and to generate new Benders cuts (13). If $\Phi_j(\hat{z}) = \hat{w}_j$, then the generated inequality does not improve the approximation to $\Phi_j(\cdot)$, and the cut is not added to (14). The Benders procedure stops when no new cuts are added. At this point, the LP relaxation of (10) is solved.

3.2. Column Generation

In our discussion to this point, we have assumed that we are given T candidate subspaces. Key to our approach is a *column generation* method for dynamically identifying new subspaces that have the potential to improve the solution to (10). Column generation is a classical method for solving large-scale LP (Ford and Fulkerson 1958) that also has seen significant use in solving MILP problems (Barnhart et al. 1998). We apply column generation to the LP relaxation of (10), or more specifically, to the Benders reformulation of this LP relaxation (14).

The key idea behind column generation is to create an auxiliary problem, called the *pricing problem*, whose solution identifies if there is an additional variable (candidate subspace), that, when added to the LP (14), could improve its solution value. The formulation of the pricing problem

follows from LP duality theory. If the *reduced cost* of a column (subspace variable) is negative, then, by increasing the value of that variable from its nominal value of zero, the objective value of the LP may decrease. Thus, we seek columns (subspaces) with negative reduced cost. If all columns have non-negative reduced cost, the current solution of the LP with the set $[T]$ of candidate subspaces is optimal and adding columns to $[T]$ would not decrease the LP relaxation value.

Consider an arbitrary subspace variable z_t. Given the optimal dual variables $(\hat{\alpha}, \hat{\beta})$ to the solution of (14), the reduced cost of a variable z_t is given by the formula

$$\frac{\lambda}{n} r_t(d - r_t) - \sum_{j \in [n]} \sum_{i \in [p_j]} \hat{\alpha}_{ji} \max\{c^*_{ji} - c_{jt}, 0\} - \hat{\beta}, \tag{15}$$

where the term $\max\{c^*_{ji} - c_{jt}, 0\}$ that is used to record the contribution of Benders cut $i \in [p_j]$ for data vector $j \in [n]$ is used to reflect the fact that the coefficient $c^*_{ji} - c_{jt}$ only appears for variable z_t in this cut when $c^*_{ji} > c_{jt}$.

We formulate the problem of finding a subspace that corresponds to a column having negative reduced cost as a problem of finding a basis matrix $U \in \mathbb{R}^{d \times r}$, where r is the candidate dimension of the subspace. In order to derive the reduced cost for the variable associated with a subspace defined by basis matrix U, we recall from (4) that for a subspace defined by basis matrix U_t,

$$c_{jt} = \Delta_{jt} + \frac{\lambda}{n} r = \min_{v \in \mathbb{R}^r} \Big\{ \sum_{i:(i,j) \in \Omega} (X_{ij} - (U_t v)_i)^2 \Big\} + \frac{\lambda}{n} r =: h_j(U_t). \tag{16}$$

Thus, substituting the function $h_j(U)$ in for c_{jt} in the reduced cost expression (15), the problem of finding a column corresponding to a rank-r basis of minimum reduced cost can be formulated as:

$$\min_{U \in \mathbb{R}^{d \times r}} g_r(U) := \frac{\lambda}{n} r(d - r) - \sum_{j \in [n]} \sum_{i \in [p_j]} \hat{\alpha}_{ji} \max\{c^*_{ji} - h_j(U), 0\} - \hat{\beta}. \tag{17}$$

The objective in problem (17) is not convex, and hence we find locally minimal solutions to (17) with a gradient-based method. To explore subspaces of varying dimensions, we solve (17) for $r \in \{1, 2, \ldots, r_{max}\}$ where r_{max} is an upper bound on the subspace dimension, which we assume is given as an input to the model.

Gradient-based method for solving (17)**.** Consider a fixed $r \in \{1, 2, \ldots, r_{max}\}$. Observe that if $h_j(U) \neq c^*_{ji}$ $\forall j \in [n], i \in [p_j]$, then the function $g_r(\cdot)$ is differentiable at U. The partial derivative of $g_r(\cdot)$ with respect to matrix element U_{ab} evaluated at current iterate $\hat{U}$ is given by

$$\frac{\partial g_r(\hat{U})}{\partial U_{ab}} = - \sum_{j \in [n]} \sum_{\substack{i \in [p_j]: \\ c^*_{ji} - h_j(\hat{U}) > 0}} 2\hat{\alpha}_{ji} \sum_{\ell \in \Omega_j} (X_{\ell j} - \hat{u}_\ell^\top \hat{v}_j) \hat{v}_{jb} \quad \forall a \in [d], b \in [r], \tag{18}$$

where $\hat{u}_\ell$ represents row ℓ of $\hat{U}$ and $\hat{v}_j$ is the optimal solution to problem (16) that is solved when evaluating $h_j(\hat{U})$ for each $j \in [n]$. By adding the condition $c^*_{ji} - h_j(\hat{U}) > 0$ in the summation,

Algorithm 1: Locally solving pricing problem for fixed subspace dimension

Data: X_Ω

Input: $U_0 \in R^{d\times r}$, maxIt=500, $\epsilon = 0.001$; `/* initial subspace */`

1 it $= 0, \hat{U} = U_0, \mathcal{U} = \{\}$; `/* iteration count */`

2 **while** it $<$ maxIt `and` $\|\nabla g_r(\hat{U})\| > \epsilon$

3 **for** $j = 1, 2, \ldots, n$ **do**

4 $\hat{v}_j = (\hat{U}_{\Omega,j}^\top \hat{U}_{\Omega,j})^{-1} \hat{U}_{\Omega,j} (X_j)_\Omega$

5 $c_{j,\text{it}} = \|(X_j)_\Omega - \hat{U}_{\Omega,j}\hat{v}_j\|_2^2 + \frac{\lambda}{n} r$

6 **end**

7 Calculate $\nabla g(\hat{U})$ using (18) ; `/* Requires` $\hat{U}, \hat{v}_j \ \forall j \in [n]$ `*/`

8 Calculate $\tilde{g}$ using (19)

9 $\gamma \leftarrow \min(0.1, \frac{\tilde{g} - g(\hat{U})}{\|\nabla g(\hat{U})\|_F^2})$; `/* Polyak step size */`

10 $\hat{U} \leftarrow \hat{U} - \gamma \nabla g(\hat{U})$; `/* gradient step */`

11 it $\leftarrow$ it $+1$ **if** $g_r(\hat{U}) < 0$

12 $\mathcal{U} \leftarrow \mathcal{U} \cup \hat{U}$

13 **end**

14 **end**

Output: Set of subspace bases $\mathcal{U}$

we implicitly use a subgradient contribution of 0 at points of non-differentiability. We denote by $\nabla g_r(\hat{U})$ the $d \times r$ matrix of partial derivatives of $g_r(\cdot)$ at $\hat{U}$.

We outline our gradient-based approach for locally solving pricing problem (17) for a fixed subspace dimension r in Algorithm 1. Since this method is not guaranteed to find a global optimal solution to the nonconvex problem (17), we run the method multiple times with different random choices of U_0 to identify different locally-optimal solutions. Our method for randomly initializing U_0 is motivated by the fact that when the matrix is fully observed, $r+1$ vectors per subspace are necessary and sufficient for subspace clustering. Hence, we randomly sample N$(> r+1)$ vectors from the M vectors in the current LP solution of (14) with the largest $\hat{w}_j$ values, where $M > N$. In our experiments we use $M = 5r_{max}$ and $N = 2r$. This approach selects a subset of vectors with high residuals in the current solution and initializes the gradient descent algorithm with a best-fit subspace on that subset of vectors. Then, we use a fast low-rank matrix completion algorithm, GROUSE (Balzano et al. 2010), to find the basis U_0 for a best-fit subspace for the sampled vectors. This U_0 is provided as an input to the Algorithm 1 (line 1).

To calculate the gradient using equation (18), we first need to solve the optimization model in (16), with $U_t = \hat{U}$, for each $j \in [n]$. The solution to this problem can be computed using the matrix $(\hat{U}_{\Omega,j}^\top \hat{U}_{\Omega,j})^{-1} \hat{U}_{\Omega,j}$ as used in line 4 (Balzano et al. 2012).

An important choice for the practical performance of gradient-based methods is the *step size*. We use the Polyak step size (Polyak 1987). The Polyak step size formula requires the optimal value of objective function, g^*. Since the optimal value g^* is unknown, we approximate it with $\tilde{g}$ (in line 8) as follows

$$\tilde{g} \approx \frac{\lambda}{n} r(d-r) - \sum_{j \in [n]} \sum_{\substack{i \in [p_j]: \\ c^*_{ji} - h_j(\hat{U}) > 0}} \hat{\alpha}_{ji} c^*_{ji}. \tag{19}$$

This choice assures that $\tilde{g}$ is a lower bound on g^*, since it replaces $h_j(U)$ in the objective of (17) with 0. The Polyak step size is then calculated as $\gamma \leftarrow \frac{\tilde{g} - g(\hat{U})}{\|\nabla g\|_2^2}$. To prevent the step size from getting too large when $\|\nabla(g)\|$ becomes small, we set $\gamma \leftarrow \min(0.1, \frac{\tilde{g} - g(\hat{U})}{\|\nabla g\|_2^2})$ (line 9). These choices were made based on preliminary empirical experiments. We discuss this in more detail in Section 4.3. The gradient step is described in line 10. We terminate when the norm of the gradient becomes sufficiently small ($\|\nabla g(\hat{U})\| \leq \epsilon = 0.001$ in our implementation) or we reach the maximum number of allowed iterations (500 in our implementation). As described in line 12, we store the basis $\hat{U}$ in each iteration if it has negative reduced cost, since each defines a potential candidate subspace to be added to the master problem (14).

3.3. MISS-DSG: Mixed Integer Subspace Selector with Dynamic Subspace Generation

We now describe how the Benders decomposition and column generation processes are integrated into an overall solution method, MISS-DSG, for the SCMD problem. Algorithm 2 describes the details. We initialize the model (14) with b randomly generated subspaces for each possible subspace dimension (line 1). In particular, to generate a subspace of dimension r, we sample a matrix from a uniform distribution over $[-1, 1]^{d \times r}$, and perform singular value decomposition (SVD) to get a basis. We then solve the master LP relaxation (14) in line 10, and generate Benders cuts for each $j \in [n]$ (lines 11-15). Observe that if $\Phi_j(\hat{z}) = \hat{w}_j$, then the generated inequality does not improve the approximation to $\Phi_j(\cdot)$, and the cut is not added to (14) (line 12). We repeat lines 11-15 until no violated cuts are found.

We next proceed to generate new columns (lines 17-27) by solving the pricing problem (17) in order to look for negative reduced cost columns. We solve the pricing problem for every candidate dimension $r \in \{1, 2, \ldots, r_{max}\}$ (line 17). For each dimension r, we do multi-start (line 18) and locally solve the pricing problem (17) using Algorithm 1 and store all bases having negative reduced cost in $\mathcal{U}$ (line 20). If any such basis are found, we add variables representing the possibility to select them to the master LP (line 22). We proceed to the next dimension if any negative reduced cost columns are found and we have exceeded the minimum number of multi-starts, $\eta_{\min}$ (lines 23-25). We perform a maximum of $\eta_{\max}$ multi-starts for each dimension r. In our experiments, we use $\eta_{\min} = 5$ and $\eta_{\max} = 15$.

Algorithm 2: Mixed Integer Subspace Selector with Dynamic Subspace Generation

Data: X_Ω

Input: max dimension r_{max}, min and max # of multi-starts: $\eta_{\min}, \eta_{max}$, max # iterations $i_{\max}$

Output: Partition of $[n]$ into K clusters: S_k, $k = 1, \ldots, K$

Initialize MILP model (10) with $[T]$ as b random subspaces of dimension r_i for each $i \in [r_{max}]$;

Calculate c_{jt} for $j \in [n], t \in [T]$;

root_node_continue $\leftarrow$ `True`, generate cuts $\leftarrow$ `True`, it $\leftarrow 0$;

while root_node_continue *and* it $< i_{\max}$

  root_node_continue $\leftarrow$ `False` ; `/* switched back on if new columns found */`

  it $\leftarrow$ it $+1$;

  `// generate Benders cuts`

  generate_cuts $\leftarrow$ `True`;

  while generate_cuts

    generate_cuts $\leftarrow$ `False` ;

    solve master LP relaxation (14) to obtain $(\hat{w}, \hat{z})$;

    for $j = 1, 2, \ldots, n$ **do**

      if $\hat{w}_j < \Phi_j(\hat{z})$

        Add Benders cuts of the form (13) to master (14);

        generate_cuts $\leftarrow$ `True`

    end

  end

  `// generate new columns`

  for $r = 1, \ldots, r_{max}$ **do**

    for $\eta = 1, \ldots, \eta_{\max}$ **do**

      Initialize U_0 with a best-fit subspace for $2r$ randomly-sampled vectors from the $5r_{max}$ vectors with largest $\hat{w}_j$;

      $\mathcal{U} \leftarrow$Solve pricing problem using Algorithm 1 to generate candidate subspaces ;

      if $\mathcal{U} \neq \emptyset$

        Extend $[T]$ to include a new z_t variables for each subspace $U_t \in \mathcal{U}$;

        if $\eta > \eta_{\min}$

          root_node_continue $\leftarrow True$;

          break ; `/* New columns found and minimum multi-starts done */`

    end

  end

  if root_node_continue

    remove all Benders cuts from (14) ; `/* invalid due to new` z_t `vars */`

end

$\hat{x}_{jt}, \hat{z}_t \leftarrow$ Solve MILP model (11) with $z_t \in \{0,1\}^T$, using a callback routine to search for violated Benders cuts at integer feasible solutions. ;

return $\{S_t = \{j \in [n] : \hat{x}_{jt} = 1\}, \forall t \in [T] \text{ s.t. } \hat{z}_t = 1\}$;

If new columns are found in the column generation process, we delete existing Benders cuts from the master problem since they become invalid due to new z_t variables (line 29), and return to the process of generating Benders cuts. We repeat cut generation (lines 8-16) and column generation (lines 17-27) as long as we find new columns having negative reduced cost or until we reach the maximum iteration limit, $i_{\max}$ which we set to be 15 in our experiments.

After we exit the root node loop we pass the updated MILP model with the new columns and cuts included to a MILP solver (line 31) to be solved by a branch-and-cut method. We do not generate new columns (z_t variables) after this point – in other words, we generate columns only at the root node, and not within the branch-and-cut phase of the solution of the MILP problem. As a result, even if we solve the pricing problem (17) to global optimality, this method is not guaranteed to find a globally optimal solution due the possibility of missing needed columns at nodes in the branch-and-bound tree. We do, however, generate Benders cuts within the branch-and-cut method for solving the MILP problem as this is necessary to assure that integer feasible solutions encountered in the solution process have their costs correctly recorded before allowing them to be accepted as an incumbent (best-known) solution. This is implemented by defining a `lazy constraint callback` that is called by the solver any time it encounters an integer-feasible solution $(\hat{w}, \hat{z})$. Within the lazy constraint callback, we check if $\hat{w}_j < \Phi_j(\hat{z})$ for any $j \in [n]$, and add the corresponding Benders cut of the form (13) to the model if so. The solver then adds these cuts to its formulation (and consequently excludes the solution that had incorrect cost value $\hat{w}_j$). After the MILP solver completes its branch-and-cut process, we use the optimal solution returned to determine the selected subspaces and mapping of each vector to a selected subspace (line 32). In particular, we select subspaces $t \in [T]$ such that $\hat{z}_t = 1$.

4. Effect of Algorithm Components

We next report results of computational studies designed to investigate the importance of various components of MISS-DSG.

4.1. Experimental Setup

We use Gurobi 8.1 for solving the LP and MILP problems and Python as the programming language. We set a time limit of 5000s for each MISS-DSG run. The computational study is conducted on a cluster of 4 core machines with a RAM of 16GB with Xeon X5690 CPU running at 3.46GHz. We report results on instances generated randomly in a fashion similar to (Lane et al. 2019). Specifically, we construct K random subspaces with bases $U_k \in \mathbb{R}^{d \times r_k}, \forall k \in [K]$ by sampling entries from a standard Gaussian distribution. We then generate n different data vectors. Each data vector $j \in [n]$ is sampled from one of the K subspaces, i.e., $X_j = U_k v_j$ for a uniform random $k \in [K]$ and $v_j \in \mathbb{R}^{r_k}$ sampled from a standard Gaussian distribution. After generating data matrix X, we

Table 1 **Effect of $|\mathbf{T}|$ on LP relaxation time (s) for $\mathbf{d=30, n=200, K=6, f=0, r_k=3\ \forall k\in[K]}$**

$\lvert T\rvert$	Without Benders	With Benders
100	0.5	0.1
500	3.2	0.3
1000	48.2	1.6
2000	729.8	2.8
4000	1380.3	3.1

Table 2 **Effect of $\mathbf{n}$ on LP relaxation time (s) for $\mathbf{d=30, K=6, f=0, |T|=500, r_k=3\ \forall k\in[K]}$**

n	Without Benders	With Benders
100	7.4	0.5
200	3.2	0.3
400	109.5	0.8
600	617.6	2.1
1200	1147.3	4.5

uniformly at random drop a percentage f of the entries in X leaving the remaining entries as the observed components Ω.

4.2. Impact of Benders Decomposition

We first demonstrate that Benders decomposition gives a very significant speedup in the time required to solve the LP in MISS-DSG . We compare the solution times for solving the LP relaxation without Benders decomposition (i.e., directly solving LP relaxation of (10)) and with Benders decomposition to solve the Benders formulation (14). For this experiment, we generate an identical set of subspaces ($[T]$) to use in the two formulations and solve the resulting LP relaxations, and we use $\lambda=0$. All results reported in this section are averaged over five different random instances for each combination of parameters.

In Table 1, we present the results for varying number of candidate subspaces ($|T|$), while fixing $d=30, n=200, K=6, f=0, r_k=3\ \forall k\in[K]$, and in Table 2, we present the results for varying n with fixed $d=30, K=6, f=0, |T|=500, r_k=3\ \forall k\in[K]$. From these two tables we observe that directly solving the LP relaxation of (8) becomes very time-consuming as either the number of candidate subspaces or the number of data points increases, whereas the time grows much more modestly when using Benders decomposition. Thus, we find that the use of Benders decomposition is essential for the computational viability of MISS-DSG .

4.3. Step-size Rule for Solving (17)

As discussed in Section 3.2, we propose to use the Polyak step size rule (Polyak 1987) when solving the pricing problem (17). A major advantage of the Polyak rule is that it is does not require tuning the initial step size, as is required when using a constant or diminishing step size rule. We illustrate this on a test instance with $\lambda=0,\ d=20, n=200, K=5, f=35\%, r_k=4\ \forall k=1,\ldots,4$. Figure 1 displays the results, where we compare the evolution of the objective obtained using Algorithm 1 with the Polyak step size and with a decaying step size ($\alpha_0/\texttt{it}$, where `it` is the iteration number). We plot the pricing problem objective value (15) on the y-axis and iteration number on x-axis. For the decaying step size rule, we consider initial step size $\alpha_0\in\{0.001, 0.01, 0.1, 1\}$. We observe that the Polyak step size leads to the fastest convergence. We found similar behavior on other test instances.

4.4. The Necessity of Multi-start

We next discuss the importance of doing multi-start when solving the pricing problem (17). We consider the same instance as discussed in Section 4.3 and solve the pricing problem with different initialization points. As shown in Figure 2, we observe that three different choices lead to three different local solutions. Hence, multi-starting can help find multiple different local minima and therefore improve the chances of identifying subspaces with negative reduced cost.

Figure 1 **Comparison of Polyak step size with decay step size for same starting point**

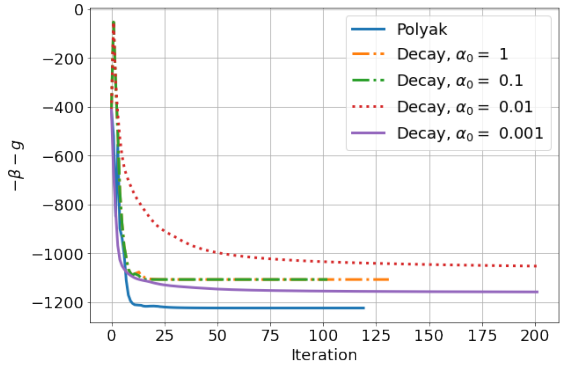
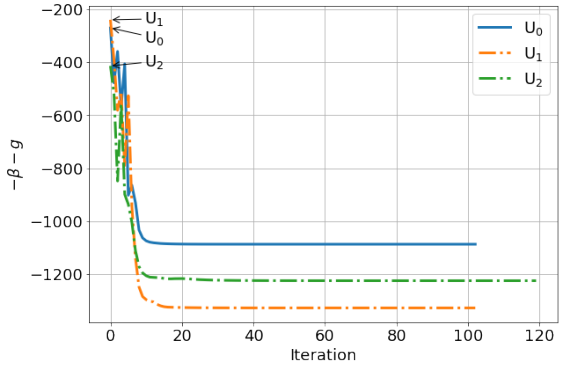


Figure 2 **Algorithm 1 converges to different local solutions for different starting points**



5. Comparison with Existing Methods

We next compare the performance of MISS-DSG against various methods from the literature.

5.1. Synthetic Dataset

The difficulty of SCMD depends on several factors such as the arrangement of subspaces, the separation between subspaces, the total dimensions of the subspaces, and the percentage of the missing data. In this section, in addition to the random instances discussed in Section 4.1, we also consider instances generated from disjoint[1] semi-random subspace arrangements. Semi-random instances allow us to control the separation between the subspaces which is measured by the affinity.

We generate intances having two and three disjoint subspaces with affinity between them being controlled by an angle parameter $\theta \in [0, \frac{\pi}{2}]$. Small values of θ indicate low affinity between the subspaces, and hence the clustering task is more challenging (Soltanolkotabi and Candés 2012). Similar to (Abdolali and Gillis 2021), we generate the two disjoint subspaces as follows:

$$U_1 = \begin{pmatrix} I_r \\ 0_r \end{pmatrix}, U_2 = \begin{pmatrix} \cos(\theta) I_r \\ \sin(\theta) I_r \end{pmatrix}$$

Here $r_1 = r_2 = r$, I_r denotes the identity matrix of size $r \times r$, and 0_r denotes the zero matrix of size $r \times r$. For an instance with n data points, we generate $\frac{n}{2}$ data points from each of the two

[1] A collection of subspaces $S_1, \ldots, S_c$ is said to be disjoint if $\dim(S_i) + \dim(S_j) = \dim(S_i \cup S_j)$ and $S_i \cap S_j = \{0\}\ \forall i \neq j$).

subspaces. We first randomly create data points within each of the two $2r$-dimensional subspaces and then transform them to the d-dimensional space. Let $\hat{X}_1$ and $\hat{X}_2$ denote data points generated from each subspace within $2r$-dimensional space. $\hat{X}_1$ is created from U_1 as $\hat{X}_1 = U_1 \times W$ where $W \in \mathbb{R}^{r \times \frac{n}{2}}$ and each entry of $W \sim \mathcal{N}(0,1)$. $\hat{X}_2$ is created similarly from U_2. These data points from dimension $2r$ are then transformed to ambient dimension d by multiplying with a randomly generated orthonormal basis $P \in R^{d \times 2r}$ as $X_i = P \times \hat{X}_i$ for $i = 1, 2$. This orthonormal projection preserves the affinity between two subspaces from the $2r$-dimensional space to the d-dimensional space.

We also create instances with three disjoint subspaces, and generate $n/3$ vectors from each of the subspaces. The three initial subspaces are constructed as (Abdolali and Gillis 2021):

$$U_1 = \begin{pmatrix} I_r \\ 0_r \end{pmatrix}, U_2 = \begin{pmatrix} \cos(\theta) I_r \\ \sin(\theta) I_r \end{pmatrix}, U_3 = \begin{pmatrix} -\cos(\theta) I_r \\ -\sin(\theta) I_r \end{pmatrix}$$

The rest of the construction is identical to the two disjoint subspace case.

After generating data matrix X either randomly or via the disjoint subspaces approach, we uniformly at random drop a percentage f of the entries of the matrix X yielding the set of observed entries Ω.

5.2. Metrics

We compare performance of all the methods in terms of clustering error and completion error defined as follows:

- Clustering Error: Let $\{G_1, G_2, \ldots, G_K\}$ be the ground truth clusters (which are known for our synthetic instances) where $G_k \subseteq [n]$ $\forall k \in [K]$ and similarly let $\{P_1, P_2, \ldots, P_{K'}\}$ be the predicted clusters. We evaluate the predicted clusters by solving the following assignment problem to find the best matching between predicted and true clusters:

$$\hat{c} := \min_{y \in \{0,1\}^{K \times K'}} \sum_{k \in [K]} \sum_{k' \in [K']} |P_{k'} \triangle G_k| y_{kk'} \tag{20a}$$

$$\sum_{k' \in [K']} y_{kk'} = 1 \qquad \forall k \in [K] \tag{20b}$$

$$\sum_{k \in [K]} y_{kk'} \leq 1 \qquad \forall k' \in [K'] \tag{20c}$$

  where binary variable $y_{kk'} = 1$ if and only if true cluster $k \in [K]$ is matched with predicted cluster $k' \in [K']$ and the objective coefficients $|P_{k'} \triangle G_k|$ measure the size of the symmetric difference between predicted cluster $P_{k'}$ and true cluster G_k. Thus, the objective (20a) minimizes the number of disagreements in the matched clusters, constraints (20b) ensure that each true cluster is mapped to exactly one predicted cluster, while constraints (20c) require that each predicted cluster is mapped to at most one true cluster. Formulation (20) is valid

when $K' \geq K$ – in case $K' < K$, the equations in (20b) are switched to $\leq$ and the inequalities (20c) are switched to equations. After solving (20), we can calculate the clustering error as $100 \times \hat{c}/2n$. Note that we divide by 2 since every vector is penalized twice when mismatched.

- Completion Error: Let Ω^c denotes the set of missing indices and I_{Ω^c} be the projection operator restricted to Ω^c. We define completion error to be the relative Frobenius distance between the true and recovered unobserved entries: $\|I_{\Omega^c} \circ (\hat{X} - X_{GT})\|_F / \|I_{\Omega^c} \circ (X_{GT})\|_F$. Here $\hat{X}$ refers to completed matrix and X_{GT} refers to the ground truth matrix. Once we recover the clusters, we perform low-rank matrix completion on the data corresponding to each cluster separately to construct $\hat{X}$. This step is done using GROUSE (Balzano et al. 2010) if the subspace dimensions are assumed known and singular value thresholding (SVT) (Cai et al. 2010) if the dimensions are unknown.

5.3. Comparison against other MIP approaches

We first benchmark MISS-DSG against the following MILP based facility-location methods proposed in the literature. These methods were proposed for the fully-observed data case. We do natural extensions to account for missing data as follows:

- FLoSS (Lazic et al. 2009): The candidate subspaces in FLoSS are initialized from the data by randomly selecting r sets of linearly independent points, with $2 \leq r < d$. Data corresponding to each set of r points defines a linear subspace of dimension $(r-1)$. The corresponding basis for fully-observed data is obtained by performing SVD on the sampled points to get the best fit subspace U. We extend this approach to partially observed data by using the same cost model as ours, i.e., the residual on observed entries (5). To handle missing data in the subspace generation process, we perform LRMC using GROUSE on each set of sampled points to get the best fit subspace. However, instead of sampling r vectors, we sample $2r$ vectors since LRMC is likely to fail with only r vectors. We then solve model (8) with $[T]$ consisting of all the subspaces generated with the above strategy. We refer to this algorithm as MIP-RANDOM.

  We point out that authors in (Lazic et al. 2009) do not specify the number of candidate subspaces ($|T|$) to construct. Since the candidate subspace generation process is random, MIP-RANDOM also serves as a good benchmark for MISS-DSG to demonstrate the value of our approach for dynamically generating subspaces. Thus, we consider a high number of candidate subspaces, $|T| = 5000$, in MIP-RANDOM whereas we initialize MISS-DSG with only 300 subspaces.
- MB-FLoSS (Lee and Cheong 2013): MB-FLoSS is similar to FloSS, with the difference being the candidate subspaces generation strategy. Instead of doing random sampling, candidate subspaces are generated by solving the following optimization problem:

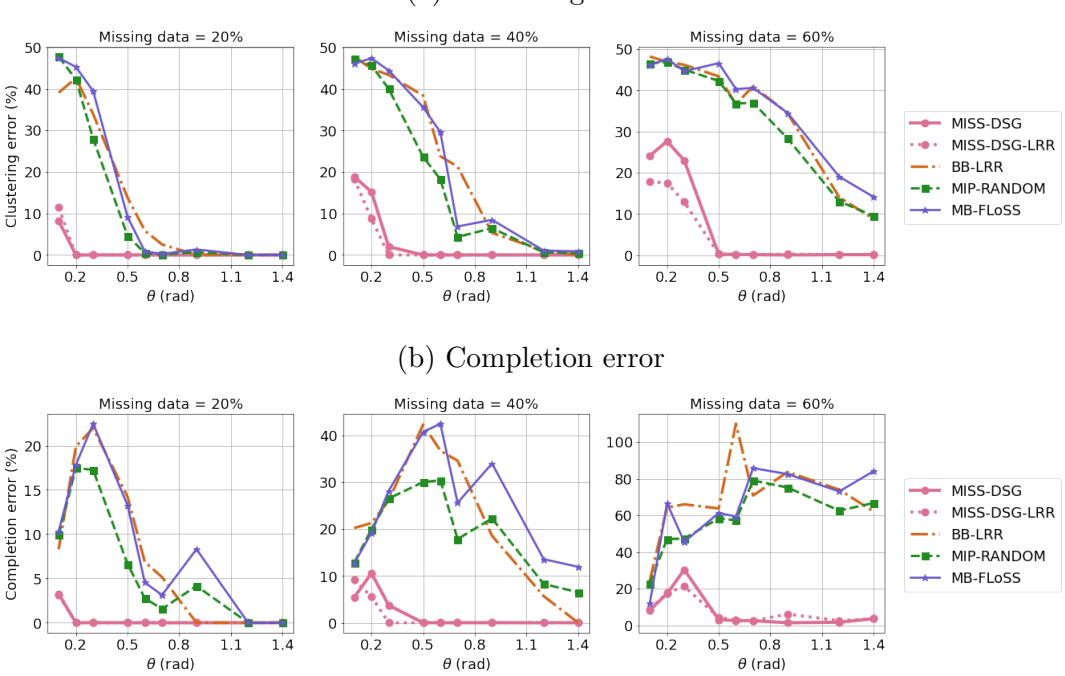
$$C^* = \arg\min_C \|C\|_{2,1} + \gamma \|E\|_{1,2} \quad \text{s.t. } X = XC + E. \tag{21}$$

Here C is the coefficient matrix of variables with $\|C\|_{2,1} := \sum_{i=1}^{d} \sqrt{\sum_{j=1}^{n} (C_{ij})^2}$, and E is the error matrix of variables with $\|E\|_{1,2} := \sum_{j=1}^{n} \sqrt{\sum_{i=1}^{d} (E_{ij})^2}$. To extend this to the missing data case, we zero-fill the missing entries when solving (21) to get C^*. Each column j of C^* represents the coefficients of other data points required to represent the data vector j. With an estimate of subspace dimension r at hand, data point j needs at most r other points for representation. Thus, for each column, we use the data points associated with the r largest coefficients in absolute value in that column to form a candidate subspace. For fully-observed data, SVD is used on these data points to get a candidate subspace. With missing data, we perform LRMC using GROUSE to get the best fit subspace which is then used as a candidate subspace. The number of candidate subspaces generated by this method is therefore the number of unique subspaces generated by considering each column of C^*.

- BB-LRR (Hu et al. 2015): BB-LRR generates candidate subspaces by over-segmentation in low rank representation (Liu et al. 2010). They set the number of clusters larger than the ground truth in the spectral clustering step, specifically they use $K+3$ instead of K. For the assignment cost function, instead of using the the residual distance directly (Δ_{jt} as in (5)), they use a normalized distance that depends on a dimension-dependent goodness of fit of subspace. The authors also suggest a randomized local method for subspace generation but we don't benchmark against it for two reasons: a) this subspace generation process is very similar to FLoSS discussed above, and b) LRR generated subspaces outperformed randomized local models in (Hu et al. 2015). We extend this algorithm to missing-data case by zero-filling the missing data during LRR step and doing LRMC with GROUSE for each cluster when generating a candidate subspace.

Disjoint subspaces: We fix parameters $d = 20$, $n = 200$, and $r_i = 2 \ \forall \ i = 1, 2$. We vary θ between $0.1 (\approx 6^\circ)$ and $1.4 (\approx 80^\circ)$. For each value of θ, we consider 10 random trials for each setting. We let the missing data percentage $f \in \{20, 40, 60\}\%$, and we report clustering and completion errors in Figure 3. For our method, we consider both a random initialization (referred to as MISS-DSG) and an initialization with LRR similar to Hu et al. (2015) (referred to as MISS-DSG-LRR). Existing facility location methods give similar performance for all three missing data cases as shown in Figure 3a. We observe that the perfect recovery threshold in terms of clustering error for existing MIP-based methods is $\theta = 0.5$ for $f = 20\%$, $\theta = 1.2$ for $f = 40\%$, and $\theta > 1.4$ for $f = 60\%$. Thus, existing MIP-based methods fail when subspaces are in close affinity or there is a high amount of missing data while MISS-DSG still gives low clustering errors in this regime. We observe a similar trend in completion error as shown in Figure 3b. For $f = 60\%$, we observe that for the existing methods the completion errors increases with θ while one expects it to decrease. The candidate subspace

Figure 3 **Performance comparison for different MIP-based methods as a function of subspace angles for two disjoint subspaces . Parameters are $\mathbf{d = 20, n = 200, K = 2}$, and $\mathbf{r_1 = r_2 = 2}$**



generation strategy in these methods use self-expressiveness and low rank matrix completion, both of which fail with high levels of missing data. The LRMC step for calculating completion error can often be faulty when an estimated cluster has points from multiple subspaces, translating to arbitrary recovery of missing entries and hence high completion errors. We also point out that MISS-DSG added between 800-4000 new candidate subspaces in these instances while MIP-RANDOM was initialized with 5000 candidate subspaces. Thus, the superior performance of MISS-DSG over MIP-RANDOM demonstrates the value of solving pricing problem (17) to generate new candidate subspaces. We also highlight that BB-LRR and MISS-DSG-LRR are initialized with the same set of initial subspaces. MISS-DSG-LRR outperforms BB-LRR by a significant margin, demonstrating that even with non-random initial subspaces, our method of generating candidate subspaces by solving pricing problem (17) leads to significant improvements.

Random subspaces: We now consider randomly generated subspaces to study the effect of missing data and ratio of ambient dimension to total rank ($d/\sum_{i=1}^{K} r_i$). We fix $d = 20, n = 240, K = 6, r_i = r = 2\ \forall i = 1, \ldots, K$, and vary the percentage of missing data between 0 to 65% as shown in Figure 4a. The reported results are averaged over 10 random trials. We observe that MISS-DSG yields low clustering error over a much wider range of missing data percentanges than the other MILP methods. We report completion errors for these instances in Table 3. MISS-DSG recovers the missing data with no completion error for f up to 50% while other MIP-based methods have

Figure 4 Performance comparison for different MIP-based methods on randomly sampled subspaces

(a) Effect of missing data on clustering error (b) Effect of d/Kr on clustering error

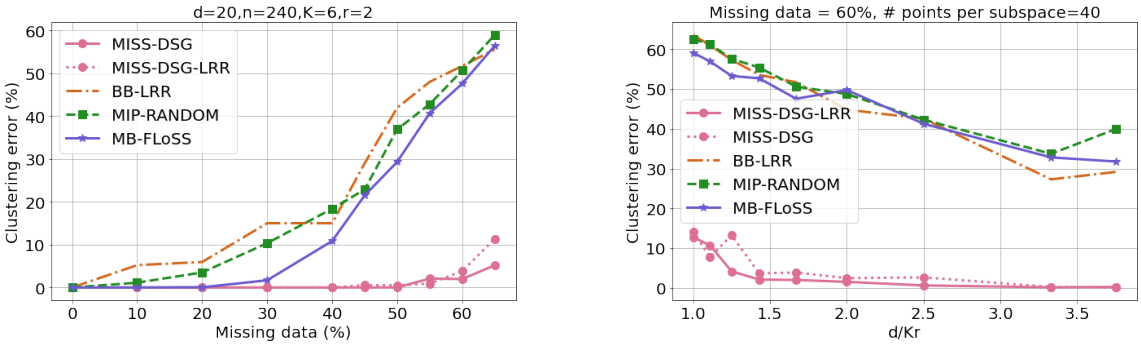


Table 3 Average completion error (%) for random instances in Figure 4a

f	BB-LRR	MIP-RANDOM	MB-FLoSS	MISS-DSG	MISS-DSG-LRR
10	28.3	2.9	0.0	0.0	0.0
20	43.6	14.5	0.6	0.0	0.0
30	116.9	43.7	17.1	0.0	0.0
40	126.2	104.3	80.7	0.0	0.0
50	237.8	198.1	173.0	0.1	6.7
55	267.4	259.7	257.7	35.2	12.7
60	324.0	311.4	295.5	41.9	43.0
65	368.7	361.3	338.5	114.5	154.4

Table 4 Average completion error (%) for random instances in Figure 4b

d/Kr	BB-LRR	MIP-RANDOM	MB-FLoSS	MISS-DSG	MISS-DSG-LRR
1.1	428.3	437.8	399.9	102.1	142.5
1.2	382.6	400.4	368.7	150.6	67.5
1.4	351.1	364.5	355.4	41.3	43.0
1.7	324.0	311.4	295.5	43.0	41.9
2.0	286.4	297.7	296.1	42.4	64.7
2.5	223.4	226.4	225.8	33.1	31.6
3.3	140.5	187.3	165.4	2.8	1.6
3.8	124.8	146.5	131.0	2.7	1.8

high completion errors for $f > 20\%$. We next study the effect of total rank of the data matrix on clustering error. We generate a variety of instances with $n/K \approx 40, f = 60\%$, and vary d, K, r to get $d/(Kr) \in [1, 4]$. A lower ratio implies that the matrix is high-rank, thus making the clustering task more difficult. Due to a high amount of missing data, we observe high clustering errors in all the existing MIP-based methods. As the matrix rank gets smaller relative to the ambient dimension, the clustering task becomes easier and hence the clustering error improves as shown in Figure 4b. MISS-DSG recovers perfect clustering when the ratio is > 2 and between $0 - 10\%$ for the ratio $\in [1, 2]$. We observe that in the high-rank and high missing data regime, low clustering error does not imply low completion errors. This is because low-rank matrix completion methods often fail in high-rank high missing data regime, leading to high completion errors, as shown in Table 4 for MISS-DSG for ratio $\in [1, 1.7]$.

5.4. Comparison against state-of-the-art

We now benchmark MISS-DSG against the following methods from the literature:

- EWZF-SSC: This is a natural extension of sparse subspace clustering to the case of missing data (Yang et al. 2015). In particular, Yang et al. (2015) proposed solving (1) with $\|\cdot\|_1$ regularization as follows:

$$C^* = \arg\min \lambda \|I_\Omega \circ (X_{ZF} - X_{ZF}C)\|_F^2 + \|C\|_1 \qquad \text{s.t. } \operatorname{diag}(C) = 0 \tag{22}$$

Coefficient matrix C^* is then processed by the spectral clustering algorithm in order to obtain data segmentation as discussed in Section 1. This method was found superior to the other methods proposed in (Yang et al. 2015) for SCMD.

- Alt-PZF-EnSC+gLRMC: In a review article on SCMD by Lane et al. (2019), alternating between elastic-net subspace clustering (You et al. 2016) and group low-rank matrix completion (Li and Vidal 2016) was found to be the state-of-the-art method. PZF is similar to EWZF and restricts error reduction on observed entries. The algorithm solves the following problem to get a coefficient matrix C^*, which is then processed by the spectral clustering algorithm to do data segmentation:

$$C^* = \arg\min \lambda \|I_\Omega \circ (X_{ZF} - X_{ZF}C)\|_F^2 + \zeta\|C\|_1 + (1-\zeta)\|C\|_F^2 \qquad \text{s.t. } \operatorname{diag}(C) = 0, \tag{23}$$

  where $0 < \zeta < 1$. The clusters obtained by spectral clustering algorithm are then processed group-wise by a low-rank matrix completion algorithm, e.g. SVT(Cai et al. 2010), to fill the missing entries and get $\hat{X}$. In next iterations, X_{ZF} in (23) is replaced by $\hat{X}$, and algorithm alternates between clustering and completion for the given number of iterations.
- k-GROUSE: Balzano et al. (2012) proposed an extension of the K-Subspaces algorithm (Bradley and Mangasarian 2000, Tseng 2000) to the case of missing data. The proposed algorithm is an alternating heuristic: starting with some initial subspaces, vectors are clustered by subspace assignment based on the same metric as (5). Given a cluster of vectors, matrix completion with Grassmannian Rank-One Subspace Estimation, GROUSE (Balzano et al. 2010), is performed to get a subspace estimate, and then vectors are reassigned, and the process is repeated until convergence. The algorithm stops when the clusters remain unchanged in successive iterations or the algorithm reaches the maximum allowed iterations

Since MISS-DSG and MISS-DSG-LRR gave similar performance, we report only MISS-DSG-LRR in these experiments. The best parameter configurations are selected based on the average completion error on a hold out set. As noted by Lane et al. (2019), this approach translates more easily into practice compared to an approach where the parameter with least classification error is selected. Being an unsupervised learning task, no true cluster labels are available in practice, so the latter approach could not be implemented. However, one can hold out some observed entries as a validation set. In our experiments, we hold out 25% of the data in the validation set for parameter selection. We report parameter choices for different algorithms in Table 5.

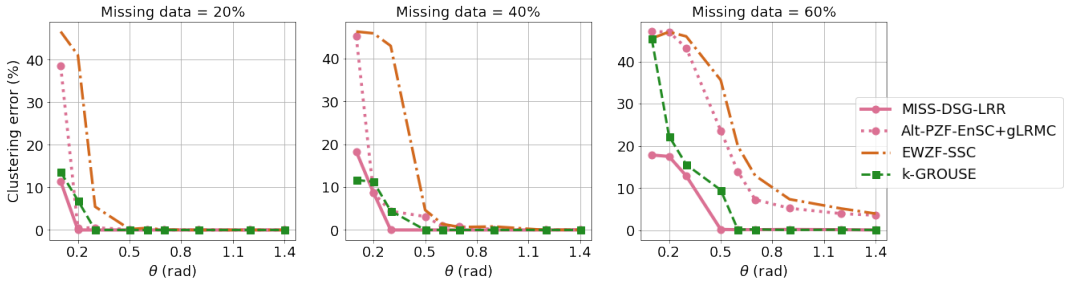
Two disjoint subspaces: We keep the same experimental setting for disjoint subspaces as in Section 5.3. We report these results in Figure 5. As expected, EWZF-SSC fails when the percentage of the missing data is high or subspaces are close to each other (small θ). MISS-DSG-LRR and k-GROUSE give the lowest clustering errors. Both of these algorithms give similar performance with

Table 5 Parameter Choices

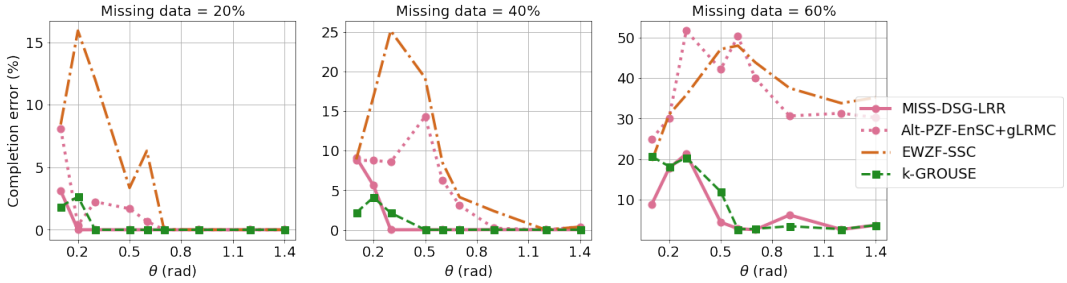
Method	Parameter
EWZF-SSC (Yang et al. 2015)	$\lambda = \frac{\alpha}{\max_{i \neq j} \|(X_{ZF})_{\Omega_j}^T (X_{ZF})_{\Omega_j})\|_{ij}}$ $\alpha \in \{5, 20, 50, 100, 200, 320\}$
k-GROUSE (Balzano et al. 2012)	-
Alt-PZF-EnSC+gLRMC (Lane et al. 2019)	λ and α similar to EWZF-SSC $\zeta \in \{0.5, 0.7, 0.9\}$

Figure 5 Performance comparison against state-of-the-art as a function of subspace angles for two disjoint subspaces. Parameters are $\mathbf{d = 20, n = 200, K = 2}$, and $\mathbf{r_1 = r_2 = 2}$

(a) Clustering error



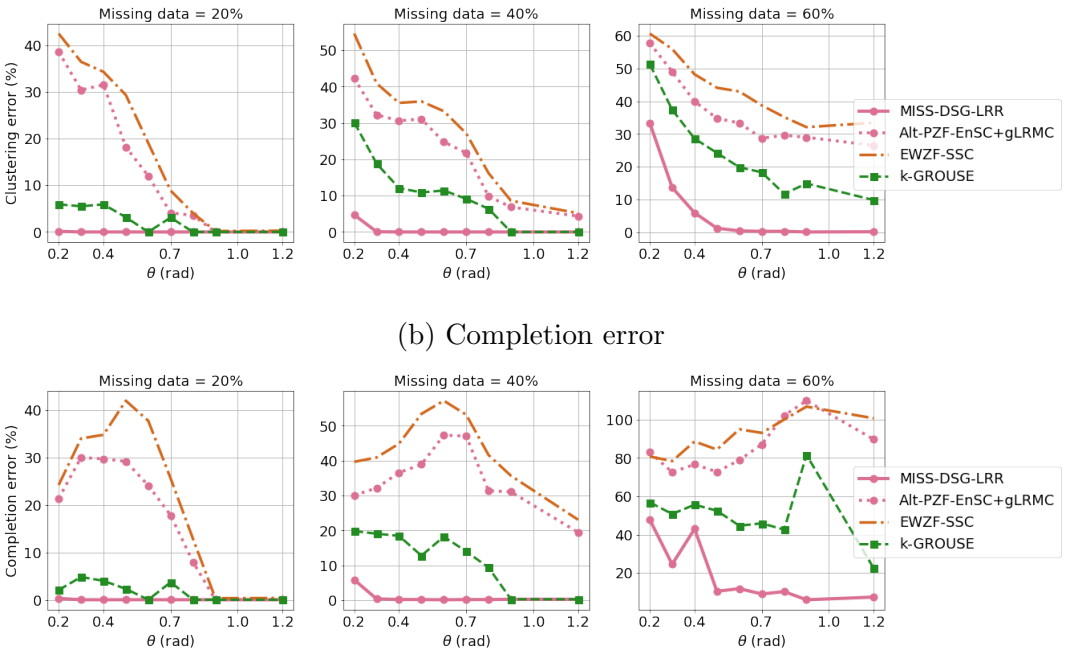
(b) Completion error



MISS-DSG-LRR doing slightly better than k-GROUSE for $f = 60\%$. A similar trend is observed in completion errors as shown in Figure 5b.

Three disjoint subspaces: We fix parameters $d = 20$, $n = 200$, and $r_k = 2 \; \forall \; k \in \{1, 2, 3\}$. We vary θ between 0.2 ($\approx 12°$) and 1.2 ($\approx 68°$). For each value of θ, we consider 10 random trials. We let missing data percentage $f \in \{20, 40, 60\}$. All methods are provided the true number of subspaces K and the true dimension of subspaces. We report these results in Figure 6. We see a significant drop in performance when compared to two disjoint subspaces for all algorithms except MISS-DSG-LRR. EWZF-SSC and Alt-PZF-EnSC+gLRMC give high clustering errors when any pair of subspaces are close to each other (small θ) or the fraction of missing data is high, and are outperformed by both k-GROUSE and MISS-DSG-LRR. Performance of k-GROUSE deteriorates in the low-affinity and high missing data regimes. MISS-DSG-LRR is the only algorithm which gives perfect recovery

Figure 6 **Performance comparison against state-of-the-art as a function of subspace angles for three disjoint subspaces. Parameters are $\mathbf{d = 20, n = 200, K = 3}$, and $\mathbf{r_1 = r_2 = r_3 = 2}$**



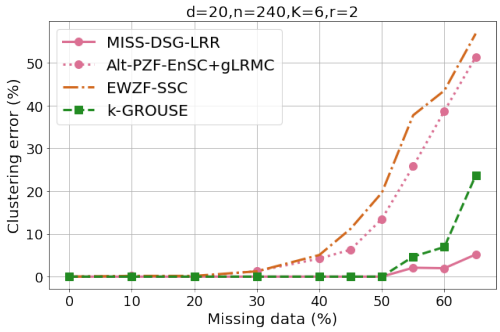
of clusters, in terms of low clustering errors as well as low completion errors, in the low-affinity and high missing data regime.

Random subspaces: For random subspaces, we fix $d = 20, n = 240, K = 6, r_i = r = 2\ \forall i = 1, \ldots, K$, and vary the percentage of missing data between 0 to 65% as shown in Figure 7a. We observe that EWZF-SSC and Alt-PZF-EnSC+gLRMC exhibit significantly high clustering errors for $f > 30\%$. k-GROUSE follows a similar trend with high clustering error for $f > 50\%$. In the high-missing data regime (40-65%), MISS-DSG -LRR yields the smallest clustering error. Completion error follows a similar trend with both EWZF-SSC and Alt-PZF-EnSC+gLRMC giving high completion error for $f > 30\%$ as shown in Table 6. MISS-DSG-LRR gives lower completion error than k-GROUSE for $f > 50\%$ while both give no completion error for $f < 50\%$.

We next study the effect of total rank (Kr) relative to the ambient dimension d on clustering error. We consider the same instances as we did in Section 5.3 for randomly sampled subspaces and vary $d/Kr \in [1, 4]$ as shown in Figure 7b. Since self-expressive methods do not perform well with high missing data (f = 60%), we find that both EWZF-SSC and Alt-PZF-EnSC+gLRMC give high clustering errors in all cases. Performance of all algorithms improve as we move from the high-rank to low-rank regime. In the high rank regime, $(1 < d/Kr < 2)$, only MISS-DSG-LRR gives near perfect classification while k-GROUSE gives errors between 5-25%. Since we have high missing data, we observe that when the matrix is nearly full-rank $(d/Kr < 2)$, all methods give

Figure 7 Performance comparison against state-of-the-art methods on a variety of synthetic instances

(a) Effect of missing data on clustering error



(b) Effect of d/Kr on clustering error

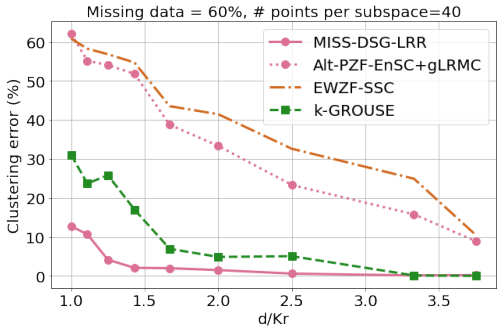


Table 6 Average completion error (%) for random instances in Figure 7a. Column 'Alt-' refers to method Alt-PZF-EnSC+gLRMC.

f	EWZF-SSC	Alt-	k-GROUSE	MISS-DSG-LRR
10	0.1	0.1	0.0	0.0
20	0.2	0.0	0.0	0.0
30	11.4	12.0	0.0	0.0
40	41.1	34.1	0.0	0.0
50	109.1	73.5	0.1	0.1
55	169.9	124.6	38.0	12.7
60	210.0	175.6	71.0	43.0
65	231.5	226.7	188.4	154.4

Table 7 Average completion error (%) for random instances in Figure 7b. Column 'Alt-' refers to method Alt-PZF-EnSC+gLRMC.

d/Kr	EWZF-SSC	Alt-	k-GROUSE	MISS-DSG-LRR
1.1	275.6	235.5	225.6	142.5
1.2	253.1	234.8	238.2	67.5
1.4	238.1	200.6	143.4	43.0
1.7	210.0	175.6	71.0	41.9
2.0	174.1	137.8	46.2	64.7
2.5	143.0	117.7	45.1	31.6
3.3	106.4	90.9	1.4	1.6
3.8	60.1	52.2	0.0	1.8

high completion errors including MISS-DSG-LRR, even though it has small clustering errors as shown in Table 7. This is due to the fact that low-rank matrix completion fails in such a setting.

5.5. Choice of penalty parameter in MISS-DSG

If we know the number of subspaces (K) and their underlying dimension r_i $\forall i = 1, \ldots, K$, then we can use $\lambda = 0$ in the objective of (10). If we do not have that information the choice of penalty parameter becomes an important hyperparameter. We conduct experiments investigating the impact of λ on instances with $d = 30, n = 300, K = 6, r = r_k = 3$ $\forall k \in [K]$, and vary $f \in \{10, 30, 50\}$. We consider three different cases for MISS-DSG:

- K known, r unknown: We use $r_{\max} = 2 * r = 6$ in Algorithm 2. Thus our model considers subspaces of dimension $\in \{1, 2, 3, 4, 5, 6\}$. However, we assume that we know number of subspaces (K), and thus we keep constraint (14c).
- K unknown, r known: MISS-DSG considers subspaces only of dimension 3. Since we don't know the number of subspaces (K), we remove constraint (14c) from our model, and let MISS-DSG self-determine the number of subspaces.
- K unknown, r unknown: We again use $r_{\max} = 2 * r = 6$ in Algorithm 2, and hence MISS-DSG considers subspaces of dimension $\in \{1, 2, 3, 4, 5, 6\}$. Similar to the previous case, we don't know

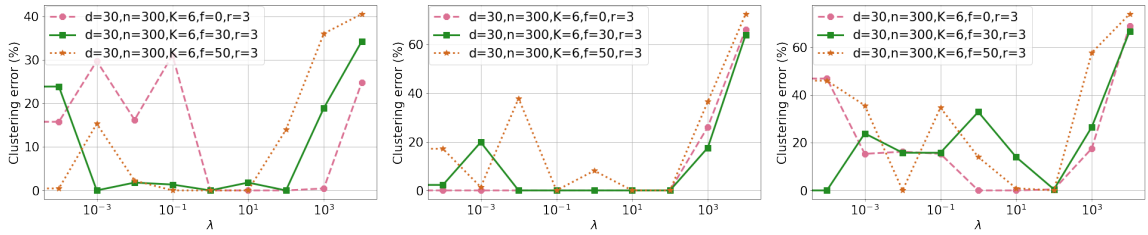
K and hence constraint (14c) is removed from the model. Thus, MISS-DSG has freedom in selecting number of subspaces as well as their dimensions.

We report the effect of λ on clustering error in Figure 8 for all three cases discussed above. We observe that MISS-DSG gives low clustering errors for a wide range of λ values when either K or r is known (Figures 8a, 8b). Extremely high values of λ lead to high clustering errors in all cases since the model is forced to either select fewer subspaces or select subspaces of lower dimension than the ground truth. For $\lambda = 10^4$, MISS-DSG selected subspaces of dimension 1. Similarly, for extremely small values of λ, MISS-DSG selects higher complexity subspaces, i.e., subspaces of dimension higher than ground truth if r is not known or higher number of subspaces than ground truth if K is unknown.

Results for the case when we do not know either r or K are shown in Figure 8c. We observe that MISS-DSG gives low clustering error only for a narrow range of λ. This behavior is expected since the model has high degree of freedom. There are multiple union of subspace models which might give low assignment cost on the observed entries ($\sum_{j\in[n]} w_j$) but have a different complexity than ground truth. Hence, choice of λ is critical in this case.

Figure 8 **Effect of λ (x axis, log scale) on clustering error**

(a) K known, r unknown (b) K unknown, r known (c) K unknown, r unknown



5.6. Hopkins155 Data Experiments

Motion segmentation has been a standard dataset in the literature for benchmarking performance of SCMD algorithms. Motion segmentation refers to the task of identifying multiple spatiotemporal regions corresponding to different rigid-body motions in a video sequence. We consider Hopkins155 motion segmentation dataset (Tron and Vidal 2007) which contains 155 video sequences with 2 or 3 moving objects. In each sequence, objects moving along different trajectories and all the trajectories associated with a single rigid motion live in a 3-dimensional affine subspace (Elhamifar and Vidal 2009). Similar to Yang et al. (2015), we subsample trajectories with six frames (equally spread) to simulate a high-rank data matrix. We handle affine subspaces in our framework by considering an affine subspace of dimension r in R^d as a linear subspace of dimension $r+1$ in R^{d+1}. Hence, we set $r_{max} = 4$ in our model. Since we do not have information on the exact dimension of the underlying

subspaces, we let our model self-determine it. However, for a fair comparison with other models, we do provide the number of subspaces as input to the model. We consider two variants of our methods: MISS-DSG with random initialization, referred to as MISS-DSG in Table 8 and MISS-DSG with initialization from Alt-PZF-EnSC+gLRMC , referred to as MISS-DSG-A in Table 8. For MISS-DSG-A, we let $\lambda = 0.1$ in our algorithm. For state-of-the-art methods, we choose the best hyperparameter as discussed in Section 5.4. We report average clustering error over 155 sequences for each method for different missing data percentage in Table 8. We observe that EWZF-SSC and k-GROUSE give similar performance with errors between 15-25% as missing data percentage is increased from 10% to 50%. MISS-DSG gives error between 18-20% for all values of missing data and is outperformed by Alt-PZF-EnSC+gLRMC in all cases. However, we observed that initializing MISS-DSG with Alt-PZF-EnSC+gLRMC generated clusters offered a great advantage. With this initialization, MISS-DSG-A was able to improve upon Alt-PZF-EnSC+gLRMC and gave errors between 5-13.9%.

Table 8 Clustering error (%) by different methods on Hopkins 155 dataset with # frames=6 and varying levels of missing data (f).

f	EWZF-SSC	k-GROUSE	Alt-PZF-EnSC+gLRMC	MISS-DSG-A	MISS-DSG
10	17.4	15.7	11.7	5.3	19.4
20	19.7	19.2	11.1	5.8	18.4
30	20.1	20.2	10.9	6.5	20.2
40	20.2	21.5	12.6	9.7	19.1
50	23.4	25.1	15.4	13.9	21.7

5.7. Computation Times

We finally discuss the computational performance of MISS-DSG. We first point out that most existing methods are faster than MISS-DSG – the advantage of MISS-DSG is its ability to successfully cluster in cases where the other methods fail. Hence, we only discuss computation times of existing methods briefly. We first discuss the computation times of existing MIP methods. MB-FLoSS and BB-LRR are significantly faster than MISS-DSG and MIP-RANDOM. Both BB-LRR and MB-FLoSS generate a small number of candidate subspaces using LRR and thus, solving the resulting MIP model is typically fast (< 2 minutes for synthetic instances considered in this paper). In fact, for these methods, the majority of the time is spent in generating clusters using LRR. For MIP-RANDOM, we sample 5000 subspaces by performing LRMC on each sampled cluster of vectors. This step took between 4-25 minutes on average for the synthetic instances considered in this work. The subspace generation time varies based on the amount of missing data since LRMC also becomes expensive for instances with high percentage of missing data. Due to a large number of candidate subspaces, solving the MIP model is also computationally more expensive than

BB-LRR and MB-FLOSS, and took between 2-10 minutes. EWZF-SSC was found to be computationally efficient on the considered synthetic instances, taking at most 2 minutes. k-GROUSE took a maximum of 12 minutes. Alt-PZF-EnSC+gLRMC is more time-consuming with computational time varying between $3 - 120$ minutes. This includes the parameter training time considered in Table 5. For Alt-PZF-EnSC+gLRMC, we had a total of 18 choices for parameters tuning. For a single parameter choice, Alt-PZF-EnSC+gLRMC gives similar computational efficiency as k-GROUSE. MISS-DSG required between $3 - 75$ minutes for the synthetic instances. We found that instances with a high-rank matrix required more computation time. Solving disjoint instances with MISS-DSG was significantly faster than the random instances, taking a maximum of 5 minutes.

We refer readers interested in more details of the computation times for MISS-DSG to the electronic companion to this paper.

6. Conclusions and future directions

We proposed a novel MILP framework MISS-DSG for the Subspace Clustering with Missing Data problem and showed it is capable of successfully clustering data in some regimes where all existing methods fail. MISS-DSG offers several other potential advantages for SCMD. It gives the user flexibility to use a different function for cost of assignment between vector and subspace. If we know a good set of potential low dimensional subspaces, our framework can take advantage of this by including these subspaces in the formulation. MISS-DSG is also capable of self-determining the number of subspaces and their dimensions, and can also easily be extended to include side constraints, e.g., ensuring that a given set of points does (or does not) lie in the same cluster. MISS-DSG is computationally more expensive than the other clustering algorithms, and we leave speed improvements such as approximate gradient calculations in pricing and parallel implementation as future work.

Acknowledgments

Support for this research was provided by American Family Insurance through a research partnership with the University of Wisconsin–Madison's Data Science Institute.

References

Abdolali M, Gillis N (2021) Beyond linear subspace clustering: A comparative study of nonlinear manifold clustering algorithms. *Computer Science Review* 42:100435, ISSN 1574-0137, .

Balzano L, Nowak R, Recht B (2010) Online identification and tracking of subspaces from highly incomplete information. *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 704–711, .

Balzano L, Szlam A, Recht B, Nowak R (2012) K-subspaces with missing data. *2012 IEEE Statistical Signal Processing Workshop (SSP)*, 612–615, .

Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, Vance PH (1998) Branch and price: Column generation for solving huge integer programs. *Operations Research* 46:316–329.

Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik* 4(1):238–252.

Bradley PS, Mangasarian OL (2000) k-plane clustering. *Journal of Global Optimization* 16(1):23–32, .

Cai JF, Candès E, Shen Z (2010) A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization* 20:1956–1982, .

Candes EJ, Tao T (2010) The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory* 56(5):2053–2080, .

Candès EJ, Recht B (2009) Exact matrix completion via convex optimization. *Foundations of Computational Mathematics* 9(6):717–772, .

Charles Z, Jalali A, Willett R (2018) Sparse subspace clustering with missing and corrupted data. *2018 IEEE Data Science Workshop (DSW)*, 180–184, .

Elhamifar E (2016) High-rank matrix completion and clustering under self-expressive models. Lee D, Sugiyama M, Luxburg U, Guyon I, Garnett R, eds., *Advances in Neural Information Processing Systems*, volume 29 (Curran Associates, Inc.), .

Elhamifar E, Vidal R (2009) Sparse subspace clustering. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2790–2797, .

Elhamifar E, Vidal R (2013) Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(11):2765–2781, .

Fan J, Chow TW (2017) Matrix completion by least-square, low-rank, and sparse self-representations. *Pattern Recognition* 71:290–305, ISSN 0031-3203, .

Fischetti M, Ljubić I, Sinnl M (2017) Redesigning benders decomposition for large-scale facility location. *Management Science* 63(7):2146–2162, .

Ford LR, Fulkerson DR (1958) A suggested computation for maximal multi-commodity network flows. *Management Science* 5(1):97–101, .

Hu H, Feng J, Zhou J (2015) Exploiting unsupervised and supervised constraints for subspace clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37(8):1542–1557, .

Huang K, Ma Y, Vidal R (2004) Minimum effective dimension for mixtures of subspaces: a robust gpca algorithm and its applications. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, II–II, .

Lane C, Boger R, You C, Tsakiris M, Haeffele B, Vidal R (2019) Classifying and comparing approaches to subspace clustering with missing data. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*.

Lazic N, Givoni I, Frey B, Aarabi P (2009) Floss: Facility location for subspace segmentation. *2009 IEEE 12th International Conference on Computer Vision*, 825–832, .

Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324, .

Lee C, Cheong L (2013) Minimal basis facility location for subspace segmentation. *2013 IEEE International Conference on Computer Vision (ICCV)*, 1585–1592 (Los Alamitos, CA, USA: IEEE Computer Society), ISSN 1550-5499, .

Li C, Vidal R (2016) A structured sparse plus structured low-rank framework for subspace clustering and completion. *IEEE Transactions on Signal Processing* 64(24):6557–6570, .

Liu G, Lin Z, Yu Y (2010) Robust subspace segmentation by low-rank representation. *Proceedings of the 27th International Conference on International Conference on Machine Learning*, 663–670, ICML'10 (Madison, WI, USA: Omnipress), ISBN 9781605589077.

Lu C, Feng J, Lin Z, Yan S (2013) Correlation adaptive subspace segmentation by trace lasso. *2013 IEEE International Conference on Computer Vision*, 1345–1352, .

Lu CY, Min H, Zhao ZQ, Zhu L, Huang DS, Yan S (2012) Robust and efficient subspace segmentation via least squares regression. Fitzgibbon A, Lazebnik S, Perona P, Sato Y, Schmid C, eds., *Computer Vision – ECCV 2012*, 347–360 (Berlin, Heidelberg: Springer Berlin Heidelberg), ISBN 978-3-642-33786-4.

Ng AY, Jordan MI, Weiss Y (2001) On spectral clustering: Analysis and an algorithm. *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, 849–856 (MIT Press).

Nguyen LT, Kim J, Shim B (2019) Low-rank matrix completion: A contemporary survey. *IEEE Access* 7:94215–94237, .

Panagakis Y, Kotropoulos C (2014) Elastic net subspace clustering applied to pop/rock music structure analysis. *Pattern Recognition Letters* 38:46–53, ISSN 0167-8655, .

Pimentel D, Nowak R, Balzano L (2014) On the sample complexity of subspace clustering with missing data. *2014 IEEE Workshop on Statistical Signal Processing (SSP)*, 280–283, .

Pimentel-Alarcón DL, Nowak R (2016) The information-theoretic requirements of subspace clustering with missing data. *ICML*.

Pimentel-Alarcón D, Balzano L, Marcia R, Nowak R, Willett R (2016) Group-sparse subspace clustering with missing data. *2016 IEEE Statistical Signal Processing Workshop (SSP)*, 1–5, .

Pimentel-Alarcón DL, Boston N, Nowak RD (2015) A characterization of deterministic sampling patterns for low-rank matrix completion. *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 1075–1082, .

Polyak B (1987) *Introduction to optimization* (Optimization Software, Inc).

Ramlatchan A, Yang M, Liu Q, Li M, Wang J, Li Y (2018) A survey of matrix completion methods for recommendation systems. *Big Data Mining and Analytics* 1:308–323, .

Rao S, Tron R, Vidal R, Yu L (2010) Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. *IEEE transactions on pattern analysis and machine intelligence* 32:1832–45, .

Recht B (2011) A simpler approach to matrix completion. *J. Mach. Learn. Res.* 12(null):3413–3430, ISSN 1532-4435.

Soltanolkotabi M, Candés EJ (2012) A geometric analysis of subspace clustering with outliers. *The Annals of Statistics* 40(4):2195–2238, ISSN 00905364, 21688966, .

Tron R, Vidal R (2007) A benchmark for the comparison of 3-d motion segmentation algorithms. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 1–8, .

Tsakiris M, Vidal R (2018) Theoretical analysis of sparse subspace clustering with missing entries. Dy J, Krause A, eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 4975–4984 (Stockholmsmässan, Stockholm Sweden: PMLR), .

Tseng P (2000) Nearest q-flat to m points. *Journal of Optimization Theory and Applications* 105(1):249–252.

Wang YX, Xu H, Leng C (2019) Provable subspace clustering: When lrr meets ssc. *IEEE Transactions on Information Theory* 65(9):5406–5432, .

Yang C, Robinson D, Vidal R (2015) Sparse subspace clustering with missing entries. Bach F, Blei D, eds., *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, 2463–2472 (Lille, France: PMLR), .

You C, Li CG, Robinson DP, Vidal R (2016) Oracle based active set algorithm for scalable elastic net subspace clustering. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3928–3937, .

Zapata EL, Gonzalez-Mora J, la Torre FD, Guil N, Murthi R (2007) Bilinear active appearance models. *2007 11th IEEE International Conference on Computer Vision*, 1–8 (Los Alamitos, CA, USA: IEEE Computer Society), .

Zhuang L, Gao H, Lin Z, Ma Y, Zhang X, Yu N (2012) Non-negative low rank and sparse graph for semi-supervised learning. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2328–2335, .