High-Rank Matrix Completion by Integer Programming

Akhilesh Soni, Jeff Linderoth, Jim Luedtke
Department of Industrial and Systems Engineering
University of Wisconsin-Madison
Madison, WI 53706
`soni6@wisc.edu` `linderoth@wisc.edu` `jim.luedtke@wisc.edu`

Daniel Pimentel-Alarcón
Department of Biostatistics and Medical Informatics
University of Wisconsin-Madison
Madison, WI 53706
`pimentelalar@wisc.edu`

Abstract

In the *High-Rank Matrix Completion* (HRMC) problem, we are given a collection of n data points, arranged into columns of a matrix $X \in \mathbb{R}^{d\times n}$, and each of the data points is observed only on a subset of its coordinates. The data points are assumed to be concentrated near a union of low-dimensional subspaces. The goal of HRMC is to recover the missing elements of the data matrix X. State-of-the-art algorithms for HRMC can fail on instances with a large amount of missing data or if the data matrix X is nearly full-rank. We propose a novel integer programming based approach for HRMC. 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 there are a small number of data points in each subspace.

1 Introduction

High-Rank Matrix Completion (HRMC) is the task of recovering the missing entries of a data matrix $X \in \mathbb{R}^{d\times n}$ whose columns, $X_1, X_2, \dots X_n$, are assumed to lie on or near a union of low-dimensional subspaces $\bigcup_{i=1}^{K} \mathcal{S}_i$, where each of the subspaces $\mathcal{S}_i$ is of dimension $r < d$. If the clustering of points is known, then the data matrix can be completed using well-known methods for low-rank matrix completion (LRMC) [1, 2, 3, 4, 5, 6, 7]. HRMC is closely related to Subspace Clustering with Missing Data (SCMD), where the goal is to identify clusters of vectors belonging to the same subspace, but components of the data vectors are missing. The SCMD and HRMC problems have applications in many areas such as image classification [8, 9], motion segmentation [10, 11], and recommendation systems [12].

Prior Work. In the last few years, many innovative new methods for SCMD and HRMC have been proposed, and the reader is directed to [13] for a more complete survey. Self-expressive methods,

Preprint. Under review.

originally proposed for complete data by Elhamifar and Vidal [14], are based on expressing each data point as a sparse linear combination of other data points. These methods have been extended to the case of missing data [15, 16, 17, 18, 19, 20]. Self-expressive methods may have trouble recovering the matrix or correctly clustering the data points when the percentage of missing data is high or the matrix is high-rank, i.e., when $Kr \approx d$. Algebraic methods that perform matrix completion without explicit clustering have also been proposed [21, 22, 23]. Algebraic methods often require a large number of points per subspace for good performance.

Another family of methods for HRMC is based on matrix factorization that directly seeks the bases of the low-dimensional subspaces [24, 25]. The resulting optimization problems are non-convex, and thus these methods are prone to converge to locally-optimal solutions. Lane et al. [13] recently did an extensive empirical evaluation of existing SCMD algorithms and concluded that zero filled sparse subspace clustering methods (based on self expressiveness) [15], when alternated with low-rank matrix completion [16], showed the overall best performance. This method is referred to as Alt-PZF-EnSC+gLRMC where Alt stands for Alternating, PZF for projected-zero filled, EnSC for elastic net subspace clustering [26], and gLRMC for group low-rank matrix completion. A disadvantage of this method is that it requires setting two regularization hyperparameters.

Mixed integer linear programming (MILP)-based methods for subspace clustering and related problems haven't been extensively explored. One exception is the work of Hu et al. [27], who give an integer programming model for subspace clustering. The model contains binary variables that assign data points to subspaces, so is similar to the model we propose in Section 2. However, the approach does not account for missing data, assumes that candidate subspaces are explicitly enumerated as input to the model, and does not scale well to large instances

Paper contributions. We propose a novel MILP solution framework for the HRMC problem that is based on dynamically determining a set of candidate subspaces and optimally assigning data points to the closest selected subspace. A key challenge in this approach is 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. Our framework can readily accommodate a huge number of candidate subspaces through its use of Benders decomposition to solve the linear programming (LP) relaxation of the MILP. The model has the advantage of integrating the subspace generation and clustering in a single, unified optimization framework. Our modeling framework also has the flexibility to include prior information about the data, if available. For example, information on a subset of points lying in the same subspace (or not) and information on lower and upper bounds on the number of subspaces or their dimensions can easily be included. Other advantages of our approach is that it does not require parameter tuning for good model performance and can easily incorporate constraints on the clusters.

As our work proposes a new method for solving an existing problem, we are not aware of potential negative societal impacts of this work.

The paper begins with a description of the integer programming formulation in Section 2. Section 3 discusses our decomposition approach to solve the model, and Section 4 presents experimental results that show the effectiveness of our framework.

2 Integer Programming Formulation

We assume that we are given a data matrix $X \in \mathbb{R}^{d \times n}$, with missing entries, whose columns are concentrated near a union of K subspaces, and each subspace is of dimension r. Note that the matrix X is low-rank when $Kr \ll \min\{d, n\}$ and high-rank when $Kr \approx \min\{d, n\}$. We let $\Omega \in \{0, 1\}^{d \times n}$ be the indicator matrix of observed entries for X, and we denote the set of integers $\{1, 2, \ldots, T\}$ as $[T]$.

Our approach is based on iteratively building a (potentially very large) collection T of candidate subspaces. Integer programming is then 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 $U_t \in \mathbb{R}^{d \times r}$ be a basis for its column subspace. We define the *closeness* c_{jt} of the vector $X_j, j \in [n]$ to a candidate subspace $t \in [T]$ as its residual (squared-distance) on the observed entries:

$$c_{jt} := \min_{v \in \mathbb{R}^r} \Big\{ \sum_{i:(i,j)\in\Omega} (X_{ij} - (U_t v)_i)^2 \Big\}. \tag{1}$$

This is a natural cost model, but different cost models could also be incorporated into our framework. However, an advantage of (1) is that it has a closed form solution in terms of a simple projection operator [24]. Specifically, 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 residual c_{jt} can be obtained as

$$c_{jt} = \|(X_j)_\Omega - P_{U_{\Omega,j}}(X_j)_\Omega\|_2^2. \tag{2}$$

Given T candidate subspaces, we formulate the HRMC problem as an integer program. Let $x_{jt} \in \{0,1\}, \forall j \in [n], t \in [T]$ be a binary assignment variable that determines if vector j is assigned to subspace t, and $z_t \in \{0,1\}, \forall t \in [T]$ be a binary selection variable that indicates whether subspace t is selected. The assignment of points to selected subspaces is similar to the facility location problem [28], where the goal is to select which facilities to open and assign each customer to one of the open facilities. In our HRMC formulation, subspaces play the role of facilities, and vectors play the role of customers. Our complete integer programming formulation is the following:

$$\min_{x,z} \sum_{t\in[T]} \sum_{j\in[n]} c_{jt} x_{jt} \tag{3a, MILP}$$

$$\sum_{t\in[T]} x_{jt} = 1, \quad \forall j \in [n] \tag{3b}$$

$$x_{jt} \le z_t, \qquad \forall j \in [n], t \in [T] \tag{3c}$$

$$\sum_{t\in[T]} z_t = K, \tag{3d}$$

$$z_t, x_{jt} \in \{0,1\}, \forall j \in [n], t \in [T]. \tag{3e}$$

The objective (3a) ensures that the model looks for the least cost assignment of vectors and subspaces. Constraint set (3b) ensures that each vector is assigned to exactly one subspace, and constraints (3c) enforce that a vector is assigned to only a selected subspace. Constraint (3d) ensures that exactly K subspaces are selected.

The MILP formulation (3) assumes that we know both the subspaces dimension (r) and the number of subspaces (K). Knowledge of the subspace dimension is fundamental to our dynamic subspace-generation approach which is based on matrix factorization. We discuss this more in detail in Section 3.2.

Casting HRMC as a facility-location type problem offers several advantages. First, the formulation can easily be extended to incorporate prior information about the data, such as vectors that should or should not belong to the same subspace, or bounds on the number of vectors assigned to a subspace. Second, the model is parameter-free, and requires no significant tuning. Finally, there has been recent significant work on solving large-scale facility location problems by exploiting their problem structure, and we can leverage these advances in our own solution approach [29].

3 Decomposition algorithm

The formulation (3) is solved via the well-known branch-and-bound method [30], which relies on solving a sequence of linear programming (LP) relaxations. The LP relaxation of (3) 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 (3). The optimal dual variables of the LP relaxation also provide a systematic mechanism for dynamically generating new candidate subspaces—a vital component of our solution framework. Because the number of candidate subspaces T and the number of points n may be quite large, solving the LP relaxation is a computational challenge. In Section 3.1, we discuss a problem-specific implementation of the Benders decomposition method for the solution of the LP relaxation to (3). Section 3.2 describes how to dynamically generate improving candidate subspaces.

3.1 Row generation

Benders decomposition is a technique that enables solution of extremely large LP problems that have special structure [31]. It has been applied to large-scale facility locations by Fischetti et al. [29], and our HRMC formulation has the same structure. The first step in the decomposition approach is a reformulation that eliminates the x_{jt} variables and adds a set of continuous variables w_j representing the assignment cost for vector $j \in [n]$. The resulting reformulation of the LP relaxation of (3) is

$$\min_{w,z} \Big\{ \sum_{j\in[n]} w_j \ : \sum_{t\in[T]} z_t = K, w_j \geq \Phi_j(z)\ \forall j \in [n], z_t \in [0,1]\ \forall t \in [T] \Big\}. \tag{4}$$

The function $\Phi_j(z)$ gives the minimum assignment cost for the vector $j \in [n]$ to a collection of subspaces parameterized by the variables $z \in [0,1]^T$. Note that the components of z may take fractional value. Specifically, $\Phi_j(z), j \in [n]$ is calculated by the following *subproblem*:

$$\Phi_j(\hat{z}) = \min_x \Big\{ \sum_{t\in[T]} c_{jt}x_t : \sum_{t\in[T]} x_t = 1, 0 \leq x_t \leq \hat{z}_t, \forall t \in [T] \Big\}. \tag{5}$$

The function $\Phi_j(z)$ is piecewise-linear and convex, and Benders decomposition works by dynamically building a lower-bounding approximation to $\Phi_j(z)$. The optimization problem (5) used to evaluate $\Phi(\cdot)$ has a closed-form solution. Moreover, its evaluation also gives sufficient information from which to create a lower-bounding approximation. Let $\{\sigma_1^j, \ldots, \sigma_T^j\}$ be a permutation of $\{1, \ldots, T\}$ satisfying $c_{j\sigma_1^j} \leq c_{j\sigma_2^j} \leq \cdots \leq c_{j\sigma_T^j}$, and let $t_j^* := \min\{t : \sum_{s=1}^t \hat{z}_{\sigma_s} \geq 1\}$ be the *critical index*. As described in [29], the *Benders cut* that can be used to lower-approximate the function $\Phi_j(\cdot)$ is

$$w_j + \sum_{i=1}^{t_j^*-1} (c_{j\sigma_{t_j^*}^j} - c_{j\sigma_i^j}) z_{\sigma_i} \geq c_{j\sigma_{t_j^*}^j}. \tag{6}$$

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

$$\min_{w,z} \sum_{j\in[n]} w_j \tag{7}$$

$$\sum_{t\in[T]} z_t = K, \tag*{(β)}$$

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

$$0 \leq z_t \leq 1, \qquad \forall t \in [T]. \tag*{(μ_t)}$$

Here β, α, 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. Solving (7) gives a solution $(\hat{w}, \hat{z})$. The subproblem (5) is then solved to evaluate $\Phi_j(\hat{z})$ for each $j \in [n]$ and to generate new Benders cuts (6). 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 (7). The Benders procedure stops when no new cuts are added. At this point, the LP relaxation of (3) is solved.

3.2 Column generation

In our discussion to this point, we have assumed that we are given T candidate subspaces. However, in reality, there are infinitely-many subspaces to consider. Let $\mathcal{T}$ be the set of all subspaces. Key to our approach is a *column generation* method for dynamically identifying new subspaces that have the potential to improve the solution to (3). Column generation is a classical method for LP [32] that also has seen significant use in solving MILP problems [33].

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 (a candidate subspace), that, when added to the LP (7), could improve its solution value. The formulation of the pricing problem follows naturally 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 could reduce. Thus, we should seek columns (subspaces) with negative reduced cost. If all columns have non-negative reduced cost, the current solution of the LP with the subset T of candidate subspaces is optimal to the true problem containing all subspaces $\mathcal{T}$.

Given the optimal dual variables (β, α) to the solution of (7), the reduced cost of a column/subspace variable z_t is given by the formula

$$-\beta - \sum_{j\in[n]} \sum_{i\in[p_j]} \alpha_{ji} \max\{c^*_{ji} - c_{jt}, 0\}, \tag{8}$$

where c_{jt} is the assignment cost of column vector X_j onto subspace t. Recall (1), that describes the assignment cost as a function of the basis matrix

$$c_{jt} := h_j(U_t) := \min_{v\in\mathbb{R}^r} \Big\{ \sum_{i:(i,j)\in\Omega} (X_{ij} - (U_t v)_i)^2 \Big\}. \tag{9}$$

Thus, to obtain a column of minimum reduced cost, we can solve the following pricing problem to identify the subspace basis matrix:

$$\max_{U\in\mathbb{R}^{d\times r}} g(U) = \sum_{j\in[n]} \sum_{i\in[p_j]} \alpha_{ji} \max\{c^*_{ji} - h_j(U), 0\}. \tag{10}$$

The problem (10) is not a convex optimization problem, and hence is difficult to solve to provable global optimality. We find locally maximal solutions to (10) with a gradient-based approach.

Gradient-based approach for pricing problem. If $h_j(U) \neq c^*_{ji}\ \forall j \in [n], i \in [p_j]$, then the function $g(U)$ is differentiable. For notational convenience, we use the indicator parameter $\hat{y}_{ji} = 1$ if $c^*_{ji} > h_j(\hat{U})$ and 0 otherwise. The partial derivative of $g(\cdot)$ with respect to matrix element U_{ab} evaluated at $\hat{U}$ is given by

$$\frac{\partial g(\hat{U})}{\partial U_{ab}} = \sum_{j\in[n]} \sum_{i\in[p_j]} 2\hat{y}_{ji}\alpha_{ji} \sum_{\ell\in\Omega_j} (X_{\ell j} - \hat{u}_\ell^\top \hat{v}_j)\hat{v}_{jb} \quad \forall a \in [d], b \in [r]. \tag{11}$$

Here $\hat{u}_\ell$ represents ℓ^{th} row of basis $\hat{U}$, and $\hat{v}_j$ is the minimizer in (9) for $U_t = \hat{U}$. If $h_j(\hat{U}) = c^*_{ji}$ for some $j \in [n], i \in [p_j]$, we can still apply the formula (11) to obtain an element of the generalized subdifferential for $g(\cdot)$ at $\hat{U}$ [34].

We outline our gradient-based approach in Algorithm 1. In each iteration, we calculate $\hat{y}$ (lines 4-6), the gradient (line 8), and move in the positive gradient direction (line 10). We use the Polyak step size [35], and this requires an estimate of the optimal value of objective function. We approximate the optimal value g^*, as $g^* \approx \sum_{j\in[n]} \sum_{i\in[p_j]} \alpha_{ji}\hat{y}_{ji}c^*_{ji}$ (line 9). Empirical experiments show that this choice of step size works well. In our implementation, we terminate the algorithm (line 2) after a maximum of 2000 iterations, if $\|\nabla g(\hat{U})\|_2 < 0.0001$, or if $g(\hat{U})$ has not improved by at least 0.01 in the last 100 iterations. Of all the columns generated, only those with negative reduced cost as calculated in (8) are added to the master problem (7).

3.3 Unified Framework

We now have all the tools to develop a unified MILP framework for HRMC problem that integrates the use of Benders decomposition and column generation. We also point out that we generate new columns (z_t variables) only at the initial LP relaxation (the so-called root node), and not at additional nodes in the branch-and-bound tree. We describe the overall MILP framework in Algorithm 2. We initialize the algorithm with m randomly generated subspaces (line 1) to initialize model (7). We then solve the master LP relaxation (7) in line 7, and generate Benders cuts for each $j \in [n]$ (lines

Algorithm 1: Gradient-based approach for locally solving pricing problem

```
Data: $X_\Omega$, subspaces dimension ($r$), critical costs $c^*_{ji}$ and dual solution $\alpha_{ji}, \forall j \in [n], i \in [p_j]$
Input: $U^0$ ;                                   /* initial subspace */
Output: Subspaces generated from each iteration
1  $\hat{U} \leftarrow U^0$ ;
2  while not converged
3      for $j = 1, 2, \ldots, n$ do
4          for $i = 1, 2, \ldots, p_j$ do
5              $\hat{y}_{ji} \leftarrow 1$ if $c^*_{ji} > \|(X_j)_\Omega - P_{\hat{U}_{\Omega,j}}(X_j)_\Omega\|_2^2$, 0 otherwise ;
6          end
7      end
8      Calculate $\nabla g(\hat{U})$ using (11) ;
9      $\hat{g} \leftarrow \sum_{j\in[n]} \sum_{i\in[p_j]} \alpha_{ji}\hat{y}_{ji}c^*_{ji}$, $\hat{\gamma} \leftarrow \frac{\hat{g}-g(\hat{U})}{\|\nabla g(\hat{U})\|_2^2}$ ;     /* Polyak step size */
10     $\hat{U} \leftarrow \hat{U} + \hat{\gamma}\nabla g(\hat{U})$ ;       /* move in positive gradient direction */
11 end
```

8-11). We repeat this until no violated cuts are found (line 10). We then proceed to generate new columns by solving the pricing problem (10). Because we use a gradient-based approach to solve the nonconvex problem (10), we initialize Algorithm 1 with different random choices of U^0 (lines 14-18) to identify different locally-optimal solutions. Each U^0 is obtained by selecting a random subset of $2r$ vectors from the $3Kr$ vectors that have the largest $\hat{w}_j$ values in the current LP solution of the master problem (7). Then, we use a fast low-rank matrix completion algorithm [5] to find the basis U^0 for a best-fit subspace for these vectors. We repeat this until we have tried at least K different U^0 and have found at least one negative reduced cost column, or reach the maximum allowed iterations. If negative reduced cost columns are found, we add them to the master LP, delete existing Benders cuts since they become invalid due to new z_t variables (lines 21-22), and return to the process of generating Benders cuts. We repeat this process as long as we are able to generate new columns.

Once we fail to find a negative reduced cost column, we exit the root node loop and pass the updated MILP model with the new columns and cuts included to a MILP solver (we use Gurobi [36]). We also provide a callback routine to the solver to generate Benders cuts as necessary when an integer solution is found during the branch-and-bound tree search. We use the solution from the MILP solver to determine the selected subspaces and assignments of vectors to them (line 27).

4 Computational study

4.1 Synthetic experiments

Experimental Setup. We construct K random subspaces with bases $U_k \in \mathbb{R}^{d\times r}, \forall k \in [K]$. Each entry of U_k is sampled from a standard Gaussian. 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 random $k \in [K]$ and $v_j \in \mathbb{R}^r$ is sampled from a standard Gaussian. We then drop a certain fraction f of the entries of the data matrix X yielding the set of observed entries Ω. We benchmark our MILP approach against zero filled sparse subspace clustering algorithm (ZF-SSC) [15], Alt-PZF-EnSC+gLRMC [13], and k-GROUSE [24]. The k-GROUSE algorithm is initialized with the output clusters from ZF-SSC and Alt-PZF-EnSC+gLRMC is initialized with output clusters from ZF-EnSC [26]. All methods are tuned for best performance with different parameters configurations that are reported in Appendix A.1. We compare the performance of MILP framework with these three algorithms to study the effect of missing data, ratio of ambient dimension to total dimension, and number of points per subspace.

We used Gurobi 8.1 as the MILP solver. Additionally, we set a time limit of 18000s (5 hours) for each MILP run. These experiments were performed on a 4 core 16 GB machine.

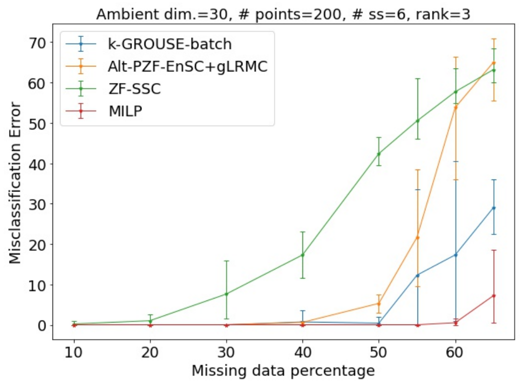
Effect of missing data fraction. We first study the effect of missing data on misclassification error. We fix $d = 30, n = 200, K = 6, r = 3$, and vary f between 10-70%. Figure 1a shows the misclassification error as a function of the missing data fraction for each algorithm. We observe

```
Algorithm 2: Unified MILP framework for SCMD
Input: $X_\Omega$, subspaces dimension $r$, number of subspaces $K$
Output: Segmentation of columns of $X$ in $K$ clusters: $S_k$ and basis $U_k, \forall k \in [K]$
1  Generate $m = 500 * K$ random subspaces to initialize MILP model (4) ;
2  root node continue ← $True$, generate cuts ← $True$;
3  while root node continue
4      root node continue ← $False$ ;      /* switched back on if new columns found */
       // generate Benders cuts
5      while generate cuts
6          generate cuts ← $False$ ;
7          solve master LP relaxation (7) to obtain $\hat{z}$;
8          for $j = 1, 2, \ldots, n$ do
9              Generate and add Benders cuts of the form (6) to master (7);
10             if cuts found: generate cuts ← $True$ ;
11         end
12     end
       // generate new columns
13     $T_n \leftarrow \emptyset$;
14     for it $= 1, \ldots,$ maxIt $= 50$ do
15         $U^0$ ← best fit subspace on randomly sampled $2 * r$ vectors from costliest $3 * K * r$
             vectors;
16         Solve pricing problem using Algorithm 1 and add columns with negative reduced cost to
             $T_n$;
17         if $|T_n| \geq 1$ and it $\geq K$: break;
18     end
19     if $T_n \neq \emptyset$
20         Calculate residual cost $c_{jt}, \forall j \in [n], t \in T_n$ using (2);
21         $[T] \leftarrow [T] \cup T_n$ ;                /* Add new $z_t, t \in T_n$ variables */
22         Remove all Benders cuts from (7) ;      /* invalid due to new $z_t$ vars */
23         root node continue ← $True$;
24     end
25 end
26 $\hat{x}, \hat{z}$ ← Solve MILP model (4) with Gurobi, give a callback routine for Benders cuts;
27 return $\{S_t = \{j \in [n] : \hat{x}_{jt} = 1\}, U_t, \forall t \in [T]$s.t. $\hat{z}_t = 1\}$ ;
```

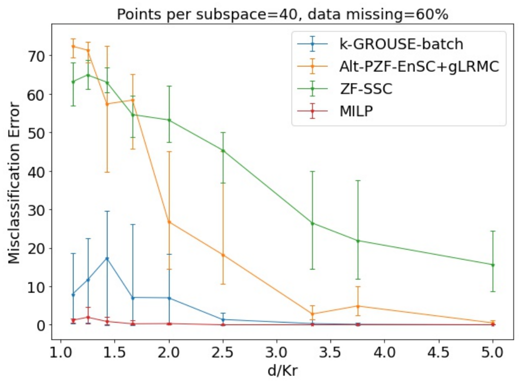
that ZF-SSC exhibits significnatly increased misclassification error already for $f > 20\%$. Alt-PZF-EnSC+gLRMC and k-GROUSE perform similarly and have increased classification error when f is larger than 50-55%. The MILP approach on the other hand successfully classifies most instances up to f around 65%. Clearly, in the high-missing data regime (40-70%), MILP yields the smallest misclassification error.

Effect of ambient dimension and total dimension of subspaces. We next study the effect of total dimension of the data ($K \times r$) relative to the ambient dimension d. We evaluate all methods in both low-rank ($d > Kr$) and high-rank ($d \approx Kr$) regimes. In these experiments, we remove $f = 60\%$ of the data and we fix $n = 40K$. We choose d, r, K such that $d/(Kr) \in [1, 5]$. We show the misclassification error for each method with respect to $d/(Kr)$ in Figure 1b. Since ZF-SSC does not perform well with high missing data ($f = 60\%$), we find that ZF-SSC gives high misclassification errors in all cases. Performance of the algorithms improve as we move towards the low-rank regime. In low-rank regime ($d/(Kr) \in [2.5, 5]$), both MILP and k-GROUSE give perfect classification and Alt-PZF-EnSC+gLRMC performs well. In the high-rank regime ($d/Kr \in [1, 2.5]$), we observe that only MILP gives near perfect classifications while Alt-PZF-EnSC+gLRMC has high misclassification. The performance of k-GROUSE lies between MILP and Alt-PZF-EnSC+gLRMC .

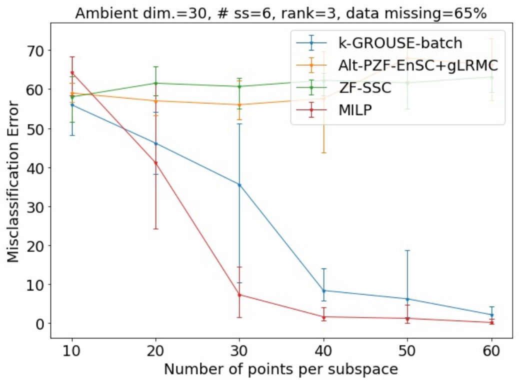
Effect of number of points per subspace. We next study the effect of the number of points per subspace. In this experiment, we fix $d = 30, n = 6, r = 3, f = 65\%$, and we vary the average number of points per subspace between 10-60. Results of this experiment are shown in Figure 1c. ZF-SSC and Alt-PZF-EnSC+gLRMC have high misclassification in all cases, since self-expressive



(a) Effect of missing data



(b) Effect of d/Kr

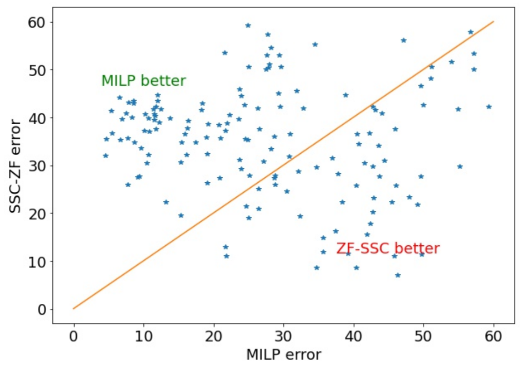


(c) Effect of points per subspace

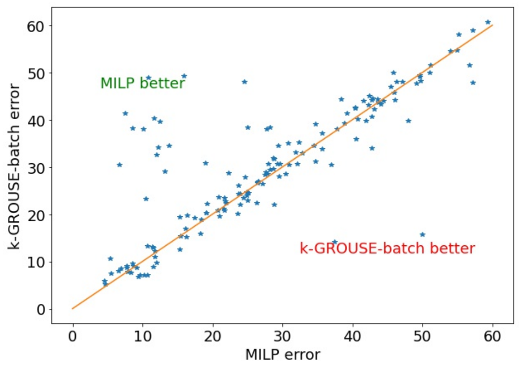
Figure 1: Misclassification error as a function of missing data, d/Kr, and # of points per subspace. Error bars plotted using minimum and maximum errors.

methods typically require more points per subspace in this high-rank regime. As the number of points increases, the accuracy of both k-GROUSE and MILP improve. MILP performs significantly better than the other methods when the number of points per subspace is in the range of 20-40. The variance in the misclassification error for the MILP is also smaller than the other algorithms as seen from the error bars in Figure 1. Residual plots for these instances are shown in Figure 3 in Appendix A.3. We also compare these methods on noisy data and found similar results (see Appendix A.4).

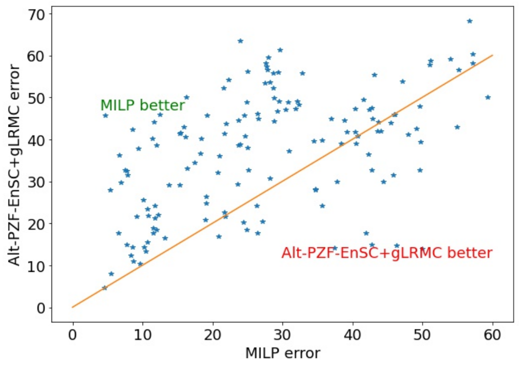
Computation Times. We report average computation times for some of the instances we considered in Section 4.1 in Table 2 in the Appendix. As expected, ZF-SSC and k-GROUSE are significantly faster than the other two methods, with computing times ranging from 2-6 minutes, whereas the computing time ranges between 45 minutes and 3.1 hours for the MILP approach and between 11 minutes and 11.4 hours for Alt-PZF-EnSC+gLRMC. For Alt-PZF-EnSC+gLRMC, the time reported includes the necessary time for hyperparameter tuning and hence is much higher than ZF-SSC since it involves solving LRMC and EnSC repeatedly for different set of parameters. We give a detailed breakdown of the computation time spent in each component of the MILP approach in Appendix A.2. A large portion of the computation in the MILP approach ($\approx 50\%$) is spent computing the c_{jt} coefficients and about 20% is spent calculating gradients, which are both operations that can be done in highly parallel fashion. Thus, while we did not pursue a parallel implementation, we expect significant speedups are possible. Overall, these results indicate that the MILP framework is feasible for moderate size data and competitive with Alt-PZF-EnSC+gLRMC, but is best suited for applications where one desires accuracy over speed, e.g., predicting gene-disease association [37].



(a) MILP vs ZF-SSC



(b) MILP vs k-GROUSE-batch



(c) MILP vs Alt-PZF-EnSC+gLRMC

Figure 2: Performance comparison of MILP with other methods on Hopkins155 dataset with 60% missing data in high-rank regime

4.2 Real Data: Motion Segmentation

We evaluate the performance of all methods on motion segmentation where we are given a video with multiple moving objects and n points are tracked across F frames. The goal is to cluster the points according to their motion subspaces. We evaluate different methods on Hopkins155, which consists of 156 different video sequences with 2 or 3 moving objects [10]. We restrict to the high-rank high-missing data regime, and hence we sample only 3 frames from each sequences since the ambient dimension d scales linearly with number of frames, i.e., $d = 2F$, and we discard $f = 60\%$ of the entries. Frames sampled are equally spaced and uniformly spread similar to [15].

Since we do not know the dimension of the underlying subspaces, we use $r = 1$ for all the methods in these experiments. We compare the performance of MILP with each algorithm individually in Figure 2. We observe that MILP gives lower misclassification errors than other methods in a majority of the sequences. When comparing MILP and ZF-SSC (Figure 2a) we see that each method performs significantly better than the other on a substantial number of instances. We suspect that MILP outperforms ZF-SSC on instances where the assumption of subspaces dimension $r = 1$ is a good approximation. When comparing MILP to k-GROUSE (Figure 2b) and Alt-PZF-EnSC+gLRMC (Figure 2c) we see that they provide similar performance on many instances, but that the number of instances where MILP is significantly better is much higher than the reverse. Therefore, MILP is a potentially better method in the high-rank and high-missing data regime. We also tested a case in which only 20% of the data entries were missing. We found Alt-PZF-EnSC+gLRMC to be clearly the best performing algorithm for this case, which is not surprising because this method does not require specification of the rank of the underlying matrix and performs well with low levels of missing data. We do not show these results in the interest of space.

5 Conclusion

We proposed a novel MILP framework for the High-Rank Matrix Completion problem and showed its effectiveness relative to other state-of-the-art methods, especially in certain instance regimes. Our MILP framework offers several other potential advantages for HRMC. 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 initial formulation. Our framework can also easily be extended to include constraints. This could be useful, for example, for fair clustering which is defined by *restricted dominance*, which limits the fraction of people from a group to be in a cluster, and *minority protection*, which requires a minimum fraction of people from a group in each cluster [38].

Our MILP approach is computationally more expensive than the other clustering algorithms. Another limitation is that given that our approach is based on matrix factorization, we require information about the dimension of the subspaces. Thus, a direction for future work is to investigate techniques for dynamically determining the dimensions of the subspaces used in the formulation. One simple idea in this direction is a sequential framework where one starts with dimension one subspaces and consecutively moves towards high dimensions while removing points which can be explained with low dimensional subspaces.

References

[1] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009. doi: 10.1007/s10208-009-9045-5.

[2] Daniel L. Pimentel-Alarcón, Nigel Boston, and Robert D. Nowak. A characterization of deterministic sampling patterns for low-rank matrix completion. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1075–1082, 2015. doi: 10.1109/ALLERTON.2015.7447128.

[3] Emmanuel J. Candes and Terence Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080, 2010. doi: 10.1109/TIT.2010.2044061.

[4] Jian-Feng Cai, Emmanuel Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20:1956–1982, 03 2010. doi: 10.1137/080738970.

[5] L. Balzano, R. Nowak, and B. Recht. Online identification and tracking of subspaces from highly incomplete information. In *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 704–711, 2010. doi: 10.1109/ALLERTON.2010.5706976.

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

[7] Luong Trung Nguyen, Junhan Kim, and Byonghyo Shim. Low-rank matrix completion: A contemporary survey. *IEEE Access*, 7:94215–94237, 2019. doi: 10.1109/ACCESS.2019.2928130.

[8] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[9] E. L. Zapata, J. Gonzalez-Mora, F. De la Torre, N. Guil, and R. Murthi. Bilinear active appearance models. In *2007 11th IEEE International Conference on Computer Vision*, pages 1–8, Los Alamitos, CA, USA, oct 2007. IEEE Computer Society. doi: 10.1109/ICCV.2007.4409185. URL `https://doi.ieeecomputersociety.org/10.1109/ICCV.2007.4409185`.

[10] Roberto Tron and Rene Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007. doi: 10.1109/CVPR.2007.382974.

[11] Shankar Rao, Roberto Tron, René Vidal, and Lei Yu. Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. *IEEE transactions on pattern analysis and machine intelligence*, 32:1832–45, 10 2010. doi: 10.1109/TPAMI.2009.191.

[12] Andy Ramlatchan, Mengyun Yang, Quan Liu, Min Li, Jianxin Wang, and Yaohang Li. A survey of matrix completion methods for recommendation systems. *Big Data Mining and Analytics*, 1: 308–323, 12 2018. doi: 10.26599/BDMA.2018.9020008.

[13] Connor Lane, Ron Boger, Chong You, Manolis Tsakiris, Benjamin Haeffele, and Rene Vidal. Classifying and comparing approaches to subspace clustering with missing data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, Oct 2019.

[14] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2765–2781, 2013. doi: 10.1109/TPAMI.2013.57.

[15] Congyuan Yang, Daniel Robinson, and Rene Vidal. Sparse subspace clustering with missing entries. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2463–2472, Lille, France, 07–09 Jul 2015. PMLR. URL `http://proceedings.mlr.press/v37/yangf15.html`.

[16] C. Li and R. Vidal. A structured sparse plus structured low-rank framework for subspace clustering and completion. *IEEE Transactions on Signal Processing*, 64(24):6557–6570, 2016. doi: 10.1109/TSP.2016.2613070.

[17] Manolis Tsakiris and Rene Vidal. Theoretical analysis of sparse subspace clustering with missing entries. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4975–4984, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/tsakiris18a.html`.

[18] Z. Charles, A. Jalali, and R. Willett. Sparse subspace clustering with missing and corrupted data. In *2018 IEEE Data Science Workshop (DSW)*, pages 180–184, 2018. doi: 10.1109/DSW.2018.8439907.

[19] Ehsan Elhamifar. High-rank matrix completion and clustering under self-expressive models. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL `https://proceedings.neurips.cc/paper/2016/file/9f61408e3afb633e50cdf1b20de6f466-Paper.pdf`.

[20] Jicong Fan and Tommy W.S. Chow. Matrix completion by least-square, low-rank, and sparse self-representations. *Pattern Recognition*, 71:290–305, 2017. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2017.05.013. URL `https://www.sciencedirect.com/science/article/pii/S0031320317302030`.

[21] Brian Eriksson, Laura Balzano, and Robert Nowak. High-rank matrix completion. In Neil D. Lawrence and Mark Girolami, editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 373–381, La Palma, Canary Islands, 21–23 Apr 2012. PMLR. URL `http://proceedings.mlr.press/v22/eriksson12.html`.

[22] Jicong Fan and Madeleine Udell. Online high rank matrix completion. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. doi: 10.1109/cvpr.2019.00889.

[23] D. Pimentel-Alarcón, G. Ongie, L. Balzano, R. Willett, and R. Nowak. Low algebraic dimension matrix completion. In *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 790–797, 2017. doi: 10.1109/ALLERTON.2017.8262820.

[24] L. Balzano, A. Szlam, B. Recht, and R. Nowak. K-subspaces with missing data. In *2012 IEEE Statistical Signal Processing Workshop (SSP)*, pages 612–615, 2012. doi: 10.1109/SSP.2012.6319774.

[25] D. Pimentel-Alarcón, L. Balzano, R. Marcia, R. Nowak, and R. Willett. Group-sparse subspace clustering with missing data. In *2016 IEEE Statistical Signal Processing Workshop (SSP)*, pages 1–5, 2016. doi: 10.1109/SSP.2016.7551734.

[26] Chong You, Chun-Guang Li, Daniel P. Robinson, and René Vidal. Oracle based active set algorithm for scalable elastic net subspace clustering. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3928–3937, 2016. doi: 10.1109/CVPR.2016.426.

[27] H. Hu, J. Feng, and J. Zhou. Exploiting unsupervised and supervised constraints for subspace clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(8):1542–1557, 2015. doi: 10.1109/TPAMI.2014.2377740.

[28] Vedat Verter. *Uncapacitated and Capacitated Facility Location Problems*, pages 25–37. Springer US, New York, NY, 2011. ISBN 978-1-4419-7572-0. doi: 10.1007/978-1-4419-7572-0_2. URL `https://doi.org/10.1007/978-1-4419-7572-0_2`.

[29] Matteo Fischetti, Ivana Ljubić, and Markus Sinnl. Redesigning benders decomposition for large-scale facility location. *Management Science*, 63(7):2146–2162, 2017. doi: 10.1287/mnsc.2016.2461.

[30] AH Land and AG Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

[31] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238–252, 1962.

[32] L. R. Ford and D. R. Fulkerson. A suggested computation for maximal multi-commodity network flows. *Management Science*, 5(1):97–101, 1958. doi: 10.1287/mnsc.5.1.97. URL `https://doi.org/10.1287/mnsc.5.1.97`.

[33] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch and price: Column generation for solving huge integer programs. *Operations Research*, 46: 316–329, 1998.

[34] Frank H. Clarke. *Optimization and Nonsmooth Analysis*. John Wiley & Sons, 1983.

[35] Boris Polyak. *Introduction to optimization*. Optimization Software, Inc, 1987.

[36] Gurobi Optimization. Algorithms in gurobi, 2016. URL `https://www.gurobi.com/pdfs/user-events/2016-frankfurt/Die-Algorithmen.pdf`.

[37] Nagarajan Natarajan and Inderjit Dhillon. Inductive matrix completion for predicting gene-disease associations. *Bioinformatics (Oxford, England)*, 30:i60–i68, 06 2014. doi: 10.1093/bioinformatics/btu269.

[38] Suman Bera, Deeparnab Chakrabarty, Nicolas Flores, and Maryam Negahbani. Fair algorithms for clustering. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/fc192b0c0d270dbf41870a63a8c76c2f-Paper.pdf`.

A Appendix

A.1 Parameters choice for state-of-the-art methods

We tuned the methods against which we compare our MILP approach for different parameter configurations. We report these configurations in Table 1. The range for each parameter is based on the recommendations from the original papers [15, 13, 24]. The best parameter configuration is selected based on the least misclassification error.

Table 1: Parameters choice for evaluated algorithms

Algorithm	**Parameters**
ZF-SSC [15]	$\lambda \in \{10^{-3}, 10^{-2}, 0.1, 1, 10, 10^2, 10^3\}$
Alt-PZF-EnSC+gLRMC [13]	$\lambda \in \{5, 50, 300\}, \gamma \in \{0.5, 0.7, 0.9\}$
k-GROUSE [24]	$\eta_0 = 0.1$, diminishing step size

A.2 Computation time

We report the total computation time of each method on a representative set of instances in Table 2.

We report detailed computational times of the MILP approach in Table 3. For the same instances reported in Table 2, we report the number of new columns generated from solving the pricing problem (# New cols), the number of Benders cuts at the Root node and in the B&B (Callback), the number of nodes explored in B&B tree, the MILP total solution time (Total sol. time), the time spent solving the pricing problem (Pricing), the time spent solving the Benders LP (Benders LP), the time spent in B&B tree (B&B), and the time spent in calculating residuals costs (Residuals). All numbers reported in Table 3 are averaged over five different random instances.

Table 2: Comparison of computation time (s). Instance notation: I-30-200-6-3-40 stands for $d = 30, n = 200, K = 6, r = 3, f = 40\%$

Instance	**ZF-SSC**	**Alt-PZF-EnSC+gLRMC**	**k-GROUSE**	**MILP**
I-40-200-4-3-60	118.6	1591.2	153.6	2680.4
I-30-200-6-3-20	163.7	672.3	181.9	3830.3
I-30-200-6-3-40	155.6	791.2	190.7	4127.3
I-30-300-6-3-65	338.9	29205.9	347.2	12299.4
I-30-360-9-3-60	290.2	41123.5	291.2	11117.7

Table 3: Detailed computational times for MILP

Instance	# New cols	# Benders cuts		# B&B nodes	Time (s)				
		Root node	Callback		Total sol. time	Pricing	Benders LP	B&B	Residuals
I-30-120-6-3-65	140241	11663	2558	11	19739.4	7944.2	126.1	755.5	6977.9
I-30-160-4-2-60	37104	7167	0	0	2002.5	998.7	7.6	2.2	997.9
I-30-160-4-3-60	49982	7071	0	0	3043.6	1426.3	9.3	3.4	1382.0
I-30-180-6-3-65	146253	14334	144	0	18751.7	8188.4	143.1	146.0	7313.3
I-30-200-5-3-60	51252	8156	0	0	2841.3	1410.3	9.2	3.2	1419.1
I-30-200-6-3-10	28246	8135	0	0	4177.5	1550.6	16.0	3.3	1736.3
I-30-200-6-3-20	33257	8998	0	0	3816.7	1440.7	30.2	2.7	1625.3
I-30-200-6-3-30	30042	10090	0	0	3417.2	1286.6	15.6	2.3	1410.0
I-30-200-6-3-40	34774	9163	0	0	4111.3	1741.6	16.4	3.9	1892.0
I-30-200-6-3-50	37211	8316	0	0	4125.6	1820.0	16.9	4.9	1937.3
I-30-200-6-3-55	50338	9613	0	0	5799.6	2663.3	19.1	4.0	2727.6
I-30-200-6-3-60	64502	10073	0	0	7058.0	3486.1	32.6	7.6	3458.6
I-30-200-6-3-65	118255	12552	120	0	14453.6	6834.9	94.9	164.8	6420.3
I-30-200-6-3-70	135352	11627	2560	11	21699.9	9593.6	129.1	2010.8	8702.6
I-30-240-6-3-60	57925	11514	0	0	4125.6	2010.9	16.4	4.1	2012.4
I-30-240-6-3-65	87303	12461	0	0	11813.3	5821.1	61.5	15.9	5598.9
I-30-280-7-3-60	71170	13946	0	0	4538.0	2169.4	22.5	7.4	2059.2
I-30-300-6-3-65	67051	15379	0	0	12272.0	5872.8	69.1	18.5	5763.1
I-30-320-8-3-60	79744	16073	188	0	15189.5	7779.2	166.9	154.7	7676.5
I-30-360-6-3-65	60075	15951	0	0	6924.0	3620.2	29.5	8.0	3603.9
I-30-360-9-3-60	86972	19349	0	0	11082.7	5647.4	72.8	15.3	5488.2
I-30-60-6-3-65	54863	5026	432	8	4114.2	1490.8	32.2	88.3	1385.7
I-40-160-4-2-60	28746	6071	28	0	4250.8	2077.7	14.6	3.2	2158.3
I-40-200-4-3-60	31498	7561	0	0	2673.6	1217.2	8.0	2.5	1286.2

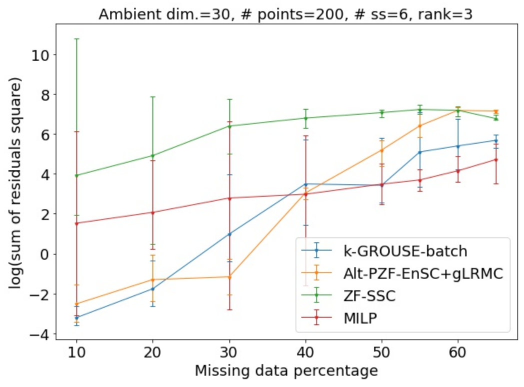
We point out that the majority of the computational time in our MILP approach is spent in calculating residuals costs and from solving the pricing problem many times. the time spent on processing the B&B tree is very small ($\approx 1\%$) of the total MILP time when no Benders cuts are generated in B&B tree with the callback. When Benders cuts are required to be generated in the B&B tree, then the time spent processing the B&B tree goes up as well. We also observe that in most cases the problem is solved without branching at all—the number of B&B nodes is 0. The time spent in solving LP relaxations while generating Benders cuts is also very small.

A.3 Residuals

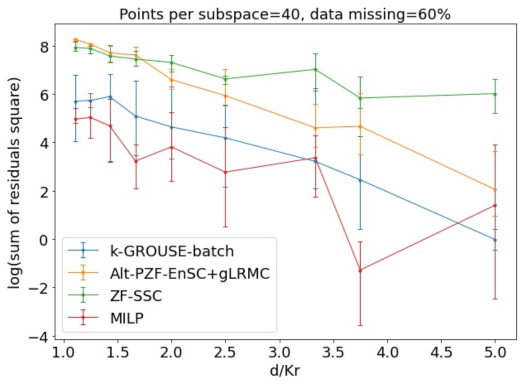
We report the sum of squared residuals from the best fit subspace for the instances reported in Section 4.1. In Figure 3, we plot the log of the sum of squared residuals on the y axis and consider different settings on the x axis, similar to Figure 1. The residual distance is calculated for each vector $j \in [n]$ from the best fit subspace (U) of the corresponding cluster: $\sum_{j\in[n]} \|(X_j)_\Omega - P_{U_{\Omega,j}}(X_j)_\Omega\|_2^2$.

We fix the values $d = 30, n = 200, K = 6, r = 3$ for the instances whose results are depicted in Figure 3a, and we observe that in low missing data regime ($< 50\%$), LRMC based methods (Alt-PZF-EnSC+gLRMC and k-GROUSE) have the smallest residuals. This is as expected; LRMC-based methods are known to perform quite well with this amount of missing data. Both Alt-PZF-EnSC+gLRMC and k-GROUSE are sensitive to the amount of missing data, and the residuals increase rapidly with an increase in missing data. MILP outperforms both of these methods in the high-missing data regime ($> 50\%$).

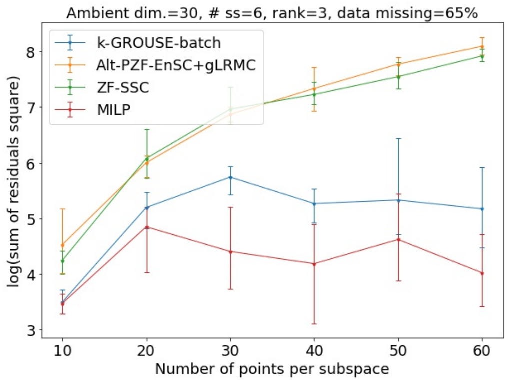
To create instances for Figure 3b, we fix $n = 40K, f = 60\%$, and we vary d, r, and K. We observe that for all the methods, log of the sum of the residuals decreases as we move towards the low-rank regime, implying that we recover a better basis for each subspace when the total dimension (Kr) is



(a) Effect of missing data



(b) Effect of d/Kr



(c) Effect of points per subspace

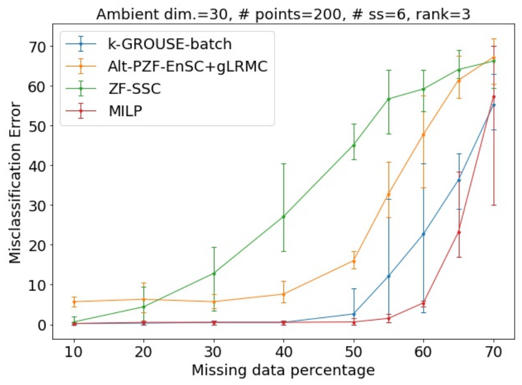
Figure 3: Sum of square of residuals from best fit subspace error as a function of missing data, d/Kr, and # of points per subspace.

smaller than ambient dimension (d). MILP yields the smallest residuals in both the low-rank and high-rank regimes.

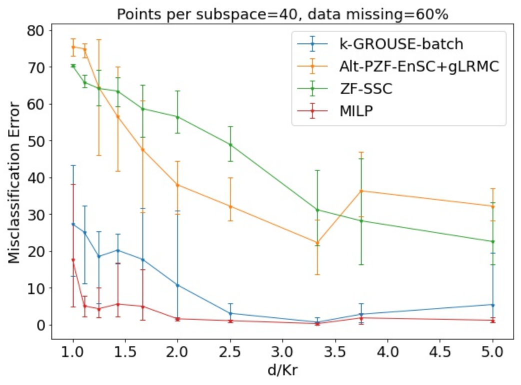
We study the effect of the average number of points sampled from each subspace in Figure 3c. To create this figure, we fix $d = 30, K = 6, r = 3, f = 65\%$, and we vary the number of points per subspace between 10-60. We observe that throughout this range, MILP yields the lowest residuals. Self-expressive methods (ZF-SSC and Alt-PZF-EnSC+gLRMC) do not perform well in these results since there are few points sampled from each subspace.

A.4 Synthetic experiments on noisy data

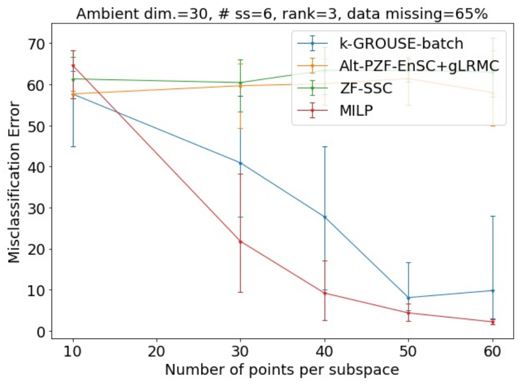
We test all the algorithms on synthetic noisy data. Similar to Section 4.1, we construct K random subspaces with bases $U_k \in \mathbb{R}^{d\times r}, \forall k \in [K]$. Each entry of U_k is sampled from a standard Gaussian. We then generate n different data vectors. Each data vector $j \in [n]$ is sampled from one of the K subspaces with additive white noise i.e. $X_j = U_k v_j + \nu_j$ for a random $k \in [K]$, $v_j \in \mathbb{R}^r$ is sampled from a standard Gaussian, and the elements of ν_j are distributed as $\mathcal{N}(0, 0.3)$. We consider the same sized instances as in Section 4.1 with additive white noise. We show the misclassification error for this setting in Figure 4 and the residuals in Figure 5. We observe that even with noise, MILP outperforms other methods in the high-missing data regime (Figures 4a and 5a), the high-rank regime (Figures 4b and 5b), and in the fewer points per subspace regime (Figures 4c and 5c).



(a) Effect of missing data

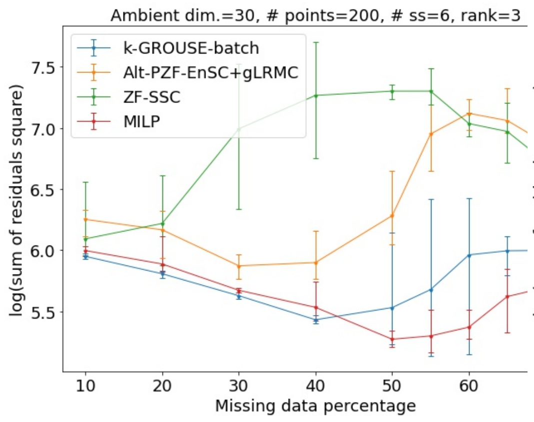


(b) Effect of d/Kr

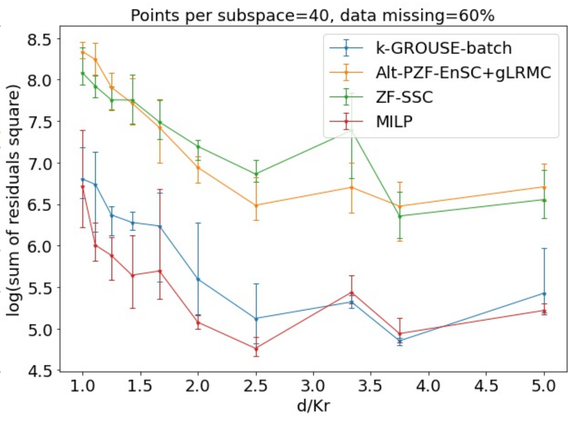


(c) Effect of points per subspace

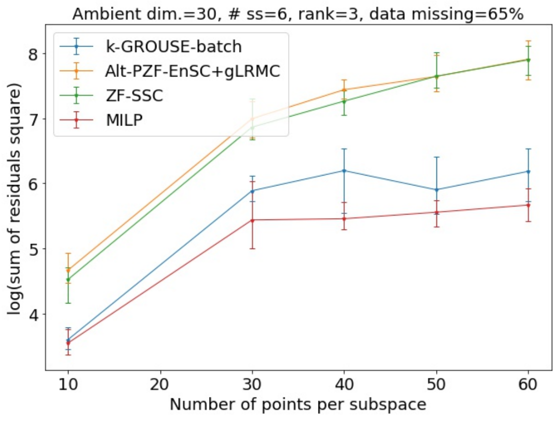
Figure 4: Misclassification error as a function of missing data, d/Kr, and # of points per subspace for noisy data



(a) Effect of missing data



(b) Effect of d/Kr



(c) Effect of points per subspace

Figure 5: Sum of square of residuals from best fit subspace as a function of missing data, d/Kr, and # of points per subspace for noisy data