DISCRETE OPTIMIZATION METHODS FOR SCHEDULING AND MATRIX COMPLETION

by

Akhilesh Soni

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Industrial and Systems Engineering)

at the

UNIVERSITY OF WISCONSIN–MADISON

2023

Date of final oral examination: 08/07/2023

The dissertation is approved by the following members of the Final Oral Committee:

Jeffrey Linderoth, Professor, Industrial and Systems Engineering
James Luedtke, Professor, Industrial and Systems Engineering
Justin J. Boutilier, Assistant Professor, Industrial and Systems Engineering
Daniel Pimentel-Alarcón , Assistant Professor, Biostatistics and Medical Informatics

To my brother, my guiding light since childhood.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisors, Jeff Linderoth and Jim Luedtke, for their unwavering support and guidance throughout my academic journey. You have been influential in teaching me how to approach a new problem and pursue meaningful inquiries. Thank you for redirecting me in the right direction every week for 5 years. I am grateful to both of you for your patience, especially during the early stages of my graduate school journey. Your passion for research and commitment to excellence have been a constant source of inspiration. Your sense of humor and approachability created a supportive and vibrant research environment that I am truly blessed to be a part of. I consider myself extremely fortunate to have had the privilege of working alongside someone as knowledgeable and experienced as both of you.

Jeff, your exceptional intelligence and sharp wit have made our interactions both intellectually stimulating and enjoyable. If I can make my presentations just 10% as engaging and enjoyable as yours someday, I would consider it a significant achievement. My best memories with you are certainly the "facet hunting" session we had in the final year of my PhD.

Jim, your brilliance and genius, coupled with your dedication to precision, has instilled in me a deep appreciation for the importance of precision in scientific inquiry. Your dedication to detail has fostered a mindset of rigor that I am sure will continue to guide me in my future endeavors. Additionally, your ability to balance your commitments to work and family serves as an inspiration and a reminder of the importance of maintaining a healthy work-life balance.

I am thankful to Justin J. Boutilier and Daniel Pimentel-Alarcón for agreeing to serve on my dissertation committee. I am also indebted to the world-class professors: Profs. Steve Wright, Jim Luedtke, Alberto Del Pia, Rob Nowak, and Eric Bach for teaching me excellent courses in the realm of mathematical optimization, algorithms, and machine learning. I am lucky to overlap with some amazing students during my time. I am thankful to Amanda Smith, Eli Towle, Rohit Kannan (postdoctoral), Rui Chen, Ramsey Rossmann, Ashley Peper, and Harshit Kothari from Luedtke Optimization (LOL) Lab, and Haoran Zhu, Zhichao Ma, and Vanessa Sawkmie from Linderoth's Lab . I am also grateful to LOL meetings for broadening my horizon and making me think about topics which I otherwise wouldn't have.

In the summers of 2019-2021, I had a chance to spend my summers working at Schneider National and Amazon. I want to thank all my friends and co-workers I met there, especially Tim Sweda, Semih Atakan, Kay Zheng, Mehdi Golari, and Guvenc Degirmenci. I would

also like to thank the excellent students I interacted with during my time in Madison outside my lab, especially Katherine Adams, Visakh Sakthidharan, Silvia Di Gregorio, Zachary Zhou, Liz Scaria, Adam Schmidt, Anjali Nair, and Kartik Sreenivasan. I am thankful to Prakarsh Pandey and Geetanjali Deole for being my family in Madison. I especially thank Geetanjali Deole for convincing me to have fun from time to time, and supporting me immensely during the last few months as I finished my dissertation.

I extend my heartfelt appreciation to my parents for teaching me the values of hard work and resilience. I am thankful to my brother, Abhishek Soni, for being my pillar of strength in both my academic and personal pursuits. To my parents, my brother, and my sister, thank you for being my biggest cheerleaders and for always believing in me. Your love and support have been instrumental in my accomplishments, and I am forever grateful for the countless sacrifices you have made for me.

The Department of Industrial and Systems Engineering at UW-Madison and Wisconsin Institute for Discovery were perfect places to spend my graduate school life, and if given a chance, I would do it all over again.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The thesis consists of research in mixed integer linear programming with applications to scheduling and matrix completion. We first study the problem of scheduling drilling and fracturing of pads in the development of an unconventional oil field. We propose a novel MILP formulation for solving this scheduling problem which considers capacity, operational, precedence, and interference constraints. We also propose a formulation that uses more decision variables, but which provides a stronger linear programming relaxation. Due to the large problem size, solving the full MILP model for instances with many pads and a large number of time periods is intractable. Thus, we also derive a MILP-based rolling horizon framework that solves a sequence of limited horizon, coarser-scale MILP instances in a rolling forward fashion to obtain a solution to the full horizon problem on the daily time scale. We benchmark this approach against a baseline scheduling algorithm that approximates current practice of scheduling pads in the order of discounted production profit with limited lookahead. Our results show that our proposed MILP-based rolling horizon approach can improve the net present value of a field by 4-6%.

Next, we present new integer programming approaches to matrix completion problems, both in the real field and in the finite field GF(2). First, we study an integer programming approach for subspace clustering with missing data problem in real field with an assumption that underlying data comes from a union of subspaces. Subspace clustering with missing data is the task of identifying clusters of vectors belonging to the same subspace in a partially observed data matrix whose columns are assumed to lie in a union of K subspaces. We propose a novel mixed-integer linear programming solution framework (MISS-DSG) for this problem that is based on dynamically determining a set of candidate subspaces and optimally assigning data points to the closest selected subspace. MISS-DSG handles a large number of candidate subspaces through its use of Benders decomposition and dynamically generates new candidate subspaces through its use of column generation. We cast the subspace generation problem as a nonlinear, nonconvex optimization problem and propose a gradient-based approximate solution approach. The model has the advantage of integrating the subspace generation and clustering in a single, unified optimization framework without requiring any hyperparameter tuning when number of subspaces and subspaces dimensions are known. Our computational results reveal that the proposed method can achieve higher clustering accuracy than state-of-the-art methods when data is of high-rank, the percentage of missing data is high, or subspaces are close to each other.

We next discuss binary matrix completion methods in GF(2) where the arithmetic is done with respect to modulo-2 operations. We give integer linear programming formula-

tions for matrix factorization and completion in GF(2). We first derive formulations making use of McCormick envelopes for the product of two binary variables: a base formulation using a general integer variable and an extended formulation using ideas from disjunctive programming and parity polytopes. The latter formulation characterizes the convex hull of the dot product of two vectors in an extended space. We then derive a novel formulation based on a new class of valid inequalities that also characterizes the convex hull of the dot product in the original space of variables. Our computational results reveal that the proposed formulation results in smaller branch-and-bound trees. Furthermore, we also derive additional classes of valid inequalities linking dot products between two matrix elements.

1.1 Overview and motivation

Mixed-integer linear programming (MILP) is a class of optimization problem where the goal is to minimize or maximize a linear objective function adhering to a set of linear constraints in presence of continuous and discrete variables. Mixed integer linear programming solvers over the last 3 decades have shown tremendous progress. CPLEX version 11.0 released in 2008 was $29530\times$ faster than version 1.2 released in 1991. Gurobi released version 1.1 in 2009 which had similar computational performance to CPLEX version 11.0 [17]. Gurobi version 10.0 released in 2022 is now $75\times$ faster compared to the version 1.1 [53]. These speedups are a combined result of many new heuristic methods, local branching, classes of new or improved cutting planes, primal heuristics, node selection, efficient preprocessing, and parallelization [62]. Along with the efficient solvers, most large scale mixed integer optimization problems are solved efficiently by exploiting the problem structure to design problem specific decomposition methods. Due to these speedups, irrespective of the fact that integer programming is *NP*-hard, integer programming has been used in practice in a diverse set of problems such as scheduling [104], supply chain design [98], telecommunication network [64], grid scheduling [86], and many others.

This progress in MILP has changed the perception that MILP is computationally intractable and cannot be used to solve machine learning problems. Recently, integer programming has been in various machine learning applications such as neural network pruning [20], binary neural networks [5], and interpretable matrix completion [15]. Moreover MILP in certain cases (e.g. clustering and classification) might allow one to solve the resulting optimization problem with an exact objective instead of a convex surrogate or a heuristic objective, the approach used in conventional machine learning algorithms to solve large problems quickly [28]. Hence, for the problems for which runtime is not an issue, formulating optimization problem as a MIP can prove to be beneficial. Consequently, MILP based models have also found applications in interpretable as well as bias correction (fairness) in machine learning [103, 69, 15].

As noted above, along with the progress in commercial solvers, research in mixed integer programming has shown that additional performance gains are often attained by adding problem specific cuts, doing careful model formulation, and using decomposition methods [60]. In this thesis, we explore methods for solving large scale MIPs for scheduling and matrix completion problems, and design algorithms which exploit the underlying structure of the specific MILP problem. We next discuss motivation for the selected problems which

is followed by review of necessary mathematical definitions.

Drill scheduling: The first problem we focus on is an engineering problem related to scheduling in unconventional oil field development. Shale oil is a type of unconventional oil found in shale formations that must be hydraulically fractured to extract the oil. Shale oil production in the U.S. has risen from 0.5 million barrels per day to 6.5 million barrels per day between 2008 and 2018, accounting to about 61% of total U.S. crude oil in 2018 production [115]. The rapid rise of shale oil production can be attributed to innovative advancements in horizontal drilling, hydraulic fracturing, and other well stimulation technologies [90]. These advanced production technologies require high level of investments and thus the strategic development of an oil field is important for the profitability of the fields. There is significant literature on conventional oil and gas infrastructure planning and development, e.g., [111, 59, 78, 50, 27] but the literature on unconventional oil and gas planning and development is more limited. We study the problem of scheduling drilling and fracturing of wells in the development of an unconventional oil field. This scheduling problem is related to the flexible flow shop scheduling problem, which has been proven to be NP-Hard [49, 52, 121]. Unfortunately, the methods for flexible flow shop scheduling problem cannot be directly applied to our problem due to the pad to pad interactions in unconventional oil field development. We propose two new MILP formulations for determining a schedule for drilling and fracturing pads and also propose a formulation that uses more decision variables, but which provides a stronger linear programming relaxation. Due to the large problem size, solving the full MILP model for instances with many pads and a large number of time periods is intractable. Thus, we also derive a MILP-based rolling horizon framework that solves a sequence of limited horizon, coarser-scale MILP instances in a rolling forward fashion to obtain a solution to the full horizon problem on the daily time scale.

Subspace clustering with missing data in $\mathbb{R}$: We next study integer programming approaches to matrix completion problem where the elements of the matrix are either real numbers (in Chapter 3) or binary values (in Chapter 4). Low-rank matrix completion problem in $\mathbb{R}$ has been well studied in literature [26, 25, 24, 7, 102, 96, 89]. Our focus in Chapter 3 is on the union of subspace model where different columns in the data matrix X lie in one of the K subspaces This is a reasonable model whenever the data vectors are generated as a linear combination of a small number of factors, often referred to as principal component vectors [9]. When data matrix X is partially observed (X_Ω), this problem is referred as *subspace clustering with missing data* (SCMD) problem. This problem has found numerous applications in computer vision [71, 124, 112, 101], recommendation systems [100], image processing [55], and systems theory [117].

State-of-the-art methods for SCMD [122, 8, 68] do not perform well when data is of

high-rank, the percentage of missing data is large, or subspaces are close to each other. The integer programming methods proposed for subspaces clustering problem with no missing data mostly rely on other algebraic and geometric methods for efficient generation of candidate subspace and use integer programming only to select best subspaces from the candidate pool [70, 72, 56]. 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 one of the open facilities. In the SCMD problem, subspaces play the role of facilities, and vectors play the role of customers. The proposed MILP based methods for SCMD are unable to handle a large number of candidate subspaces and do not fully exploit the capabilities of mixed integer linear programming tools. We bridge this gap by proposing MISS-DSG : Mixed Integer Subspace Selector with Dynamic Subspace Generation. MISS-DSG handles a large number of candidate subspace through its use of Benders decomposition and dynamically generates new candidate subspaces through its use of column generation.

Low-rank matrix factorization and union of subspace model in $\mathbb{F}_2$: We then transition to the binary setting where additions and multiplications are with respect to the finite field $\mathbb{F}_2$, which follows modulo-2 arithmetic. The problem of reconstructing X from a partially observed matrix X_Ω with its entries in $\mathbb{R}$ has been extensively studied [89]. However, the problem has received less attention in the case of finite fields. Nonetheless, reconstructing X and factorizing X into binary matrices ($X = UV$) has important applications in network and index coding, independent component analysis, social networks, market-based data, DNA transcription profiles, and others [76, 66]. For the binary setting with arithmetic defined over $\mathbb{F}_2$, we consider both matrix factorization and completion models. Matrix factorization and completion in $\mathbb{F}_2$ are known to be *NP*-hard [54]. Due to its relationship to the linear index coding problem, heuristic methods have been proposed in the network coding literature over the last few years [11, 40, 58]. Excluding the work of Saunderson et al. [106], irrespective of the underlying combinatorial and discrete nature of the problem, no linear or integer programming based methods have been studied. Saunderson et al. [106] proposed a LP-based heuristic algorithm. Unfortunately, this relaxation is quite weak, keeping only a small set of linear inequalities valid for the original integer program. We bridge this gap by studying integer programming methods for matrix factorization and completion in $\mathbb{F}_2$.

We next review important concepts from linear programming, polyhedral theory, integer programming, and linear algebra that are relevant to this thesis.

1.2 Mathematical background

Let $c \in \mathbb{R}^{n_1}, d \in \mathbb{R}^{n_2}, A \in \mathbb{R}^{n_1 \times m}, B \in \mathbb{R}^{n_2 \times m}, b \in \mathbb{R}^m$. A general mixed-integer linear program can then be formulated as follows:

$$\begin{aligned} &\min c^T x + d^T y \\ &Ax + By = b \\ &x \in \mathbb{R}^{n_1}, y \in \mathbb{Z}^{n_2} \end{aligned} \tag{MILP}$$

If $n_1 = 0$, i.e., no continuous variables, the above program reduces to pure integer program, and if $n_2 = 0$, i.e., no integer variables, then it reduces to a linear program (LP). The set $\mathcal{S}$ of feasible solutions to this mixed-integer linear program is called a mixed integer linear set. We refer to the set of feasible solutions to above program as $\mathcal{S}$ and convex hull of feasible solutions as conv($\mathcal{S}$). conv($\mathcal{S}$) is a polyhedron and solving above MILP is equivalent to

$$\begin{aligned} &\min c^T x + d^T y \\ &(x, y) \in \text{conv}(\mathcal{S}) \end{aligned}$$

1.2.1 Polyhedra theory

Definition 1.1 (Affine independence)**.** Points $x_1, x_2, \ldots, x_d \in \mathbb{R}^n$ are affinely independent if there does not exist $\lambda \in \mathbb{R}^d$ such that

$$\sum_{i=1}^{d} \lambda_i = 1, \sum_{i=1}^{d} \lambda_i x_i = \mathbf{0}$$

Definition 1.2 (Dimension of a set)**.** Given a nonempty set $X \in \mathbb{R}^n$, the dimension of X dim(X) is the largest integer d such that there exists $x_1, x_2, \ldots, x_{d+1} \in X$ which are affinely independent.

Definition 1.3 (Halfspace)**.** A hyperplane in $\mathbb{R}^n$ is defined as the set of point satisfying $ax \leq b$ for some $a \in \mathbb{R}^n, b \in \mathbb{R}$.

Definition 1.4 (Polyhedron)**.** A polyhedron $\mathcal{P}$ in $\mathbb{R}^n$ is defined as the intersection of finitely many halfspaces. In other words, $\mathcal{P} := \{x : Ax \leq b\}, A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$. A bounded polyhedron is called *polytope*.

Definition 1.5 (Convex hull)**.** Convex hull of set $\mathcal{S}$ is

$$\text{conv}(\mathcal{S}) = \{\sum_{i=1}^{p} \lambda_i x_i : i \in \mathbb{N}, x_i \in S \; \forall i \in [k], \lambda \in \mathbb{R}^k_+, \sum_{i=1}^{p} \lambda_i = 1\}$$

Definition 1.6 (Valid inequality)**.** An inequality $\alpha^\top x \leq \beta$ is valid for a set $X \subset \mathbb{R}^n$ if $\alpha^\top y \leq \beta$ for any $y \in X$.

Definition 1.7 (Face)**.** Set $F \subseteq \mathbb{R}^n$ is called a face of polyhedron $\mathcal{P} := \{x \in \mathbb{R}^n : Ax \leq b\}$ if there exists $\alpha \in \mathbb{R}^n, \beta \in \mathbb{R}$ such that $\alpha^\top x \leq \beta$ is valid for $\mathcal{P}$ and

$$F = \mathcal{P} \cap \{x \in \mathbb{R}^n : \alpha x = \beta\}.$$

A nonempty face F of $\mathcal{P}$ is called a *facet* if and only if dim(F)=dim(P)-1.

Definition 1.8 (Projections)**.** The orthogonal projection of set $\mathcal{S} \subset \mathbb{R}^{n_1+n_2}$ onto the linear subspace $\mathbb{R}^{n_1}$ is $\pi_x(\mathcal{S}) := \{x \in \mathbb{R}^{n_1} : \exists z \in \mathbb{R}^{n_2} \text{ s.t. } (x, z) \in \mathcal{S}\}$.

1.2.2 Lifting valid inequalities

We let $B := \{x \in \{0,1\}^n : Ax \leq b\}$ be a binary set and $C \subseteq [n]$ be the index set. We next present some important results related to lifting valid inequalities for the binary set S.

Definition 1.9 (Lifting)**.** For a binary set B, index set C, and a valid inequality $\sum_{j \in C} \alpha_j x_j \leq \beta$ for $\text{conv}(B) \cap \{x \in R^n : x_j = 0, j \notin C\}$, an inequality $\sum_{j \in [n]} \alpha_j x_j \leq \beta$ is called a *lifting* of $\sum_{j \in C} \alpha_j x_j \leq \beta$ if it is valid for $\text{conv}(B)$.

We next state a useful result for lifting from [30].

Proposition 1.10 (Proposition 7.2, Conforti et al. [30])**.** *Consider a set $B \subseteq \{0,1\}^n$ such that $B \cap \{x : x_n = 1\} \neq \emptyset$, and let $\sum_{i=1}^{n-1} \alpha_i x_i \leq \beta$ be a valid inequality for $B \cap \{x : x_n = 0\}$. Then*

$$\alpha_n := \beta - \max \left\{ \sum_{i=1}^{n-1} \alpha_i x_i : x \in B, x_n = 1 \right\}$$

is the largest coefficient such that $\sum_{i=1}^{n-1} \alpha_i x_i + \alpha_n x_n \leq \beta$ is valid for B.

Furthermore, if $\sum_{i=1}^{n-1} \alpha_i x_i \leq \beta$ defines a d-dimensional face of $\text{conv}(B) \cap \{x_n = 0\}$, then $\sum_{i=1}^{n} \alpha_i x_i \leq \beta$ defines a face of $\text{conv}(B)$ of dimension at least $d + 1$.

We also note that Proposition 1.10 can be extended to the case when $\sum_{i=1}^{n-1} \alpha_i x_i \leq \beta$ is a valid inequality for $B \cap \{x : x_n = 1\}$ by introducing a new variable $x'_n = 1 - x_n$, and then $\sum_{i=1}^{n-1} \alpha_i x_i \leq \beta$ is a valid inequality for $B \cap \{x : x'_n = 0\}$.

Sequential lifting [30] We next discuss *sequential lifting* which is a procedure to lift a facet-defining inequality $\sum_{j\in C}\alpha_j x_j \leq \beta$ of $\mathrm{conv}(B)\cap\{x : x_j = 0, j\notin C\}$ into a facet-defining inequality $\sum_{j=1}^{n}\alpha_j x_j \leq \beta$ of $\mathrm{conv}(B)$

Choose an ordering $j_1,\ldots,j_\ell$ of the indices in $[n]\backslash C$. Let $C_0 = C$ and $C_h = C_{h-1}\cup\{j_h\}$ for $h = 1,\ldots,\ell$. For $h = 1$ up to $h = \ell$, compute

$$\alpha_{j_h} := \beta - \max\left\{\sum_{j\in C_{h-1}}\alpha_j x_j : x\in B, x_j = 0, j\in[n]\backslash C_h, x_{j_h} = 1\right\}.$$

By Proposition 1.10, the inequality $\sum_{j=1}^{n}\alpha_j x_j \leq \beta$ obtained this way is facet-defining for $\mathrm{conv}(B)$.

1.2.3 Algorithms for solving MILPs

Branch-and-bound method lies at the heart of modern integer programming solvers. Branch and bound solution methodology is based on optimizing the objective function on different partitions of the feasible set $\mathcal{S}$. Branching involves creating these partitions by performing linear disjunctions. For bounding the optimal value, although there are several procedures, the most common is to form the linear programming relaxation.

Obtaining good bounds on MILPs can aid branch and bound procedure. These bounds are obtained by efficient polyhedral approximations of $\mathrm{conv}(\mathcal{S})$. If *convex hull*, $\mathrm{conv}(\mathcal{S})$, of solution set $\mathcal{S}$ is characterized exactly, then MILP reduces to a linear program and no branching is needed. The difficulty of course lies in characterizing $\mathrm{conv}(\mathcal{S})$. Nonetheless, close approximations to $\mathrm{conv}(\mathcal{S})$ can lead to smaller branch and bound trees.

Tightening continuous relaxing of MILPs

To solve a MILP, one typically starts with an approximation to $\mathrm{conv}(\mathcal{S})$ (e.g., LP relaxation). The continuous approximation obtained by relaxing the integrality constraints, barring some special cases, is often weak. Additional polyhedral information can help to augment this approximation dynamically. Cutting plane and column generation methods have proven to be powerful techniques for achieving this feat. Both cutting plane and column generation alternate between computing bound information by solving a "master problem" and improving the approximation to $\mathrm{conv}(\mathcal{S})$ by solving a "subproblem".

Cutting plane Cutting plane methods generate half-spaces that contain $\mathrm{conv}(\mathcal{S})$ but not the linear relaxation. These halfspaces are commonly referred to as valid inequalities and are added iteratively to the current approximation of $\mathrm{conv}(\mathcal{S})$. Valid inequalities are

generated by solving a subproblem, referred to as *separation problem*, which uses the primal solution information as an input. A typical workflow for a *pure cutting plane* approach without branching is as follows:

1. Solve the LP relaxation of MILP by relaxing the integrality constraints
2. If LP relaxation is infeasible or unbounded, STOP. MILP is also infeasible or unbounded.
3. If not, let (x, y) be an optimal solution the LP relaxation.
   - If $(x, y) \in \mathcal{S}$, STOP. (x, y) is an optimal solution.
   - If $(x, y) \notin \mathcal{S}$, solve the *separation problem* using (x, y) as an input. Add generated valid inequality to the LP relaxation and go to step 1.

In often cases, cutting plane procedure is employed within branch-and-bound method to to improve the bounds found via the LP relaxation.

Column generation Column generation technique is an iterative procedure applied to linear programs with exponential number of variables. Column generation starts by solving *restricted master problem* (RMP), in which only few variables (columns in constraint coefficient matrix) are considered. New columns with negative reduced costs are generated iteratively and added to the RMP by solving a subproblem referred to as *pricing subproblem*. Column generation can also be thought of as dynamically generating the extreme points and adding them to the current approximation to conv$(\mathcal{S})$ to generate an improved approximation.

Cutting plane methods shrink the current polyhedron approximation of conv$(\mathcal{S})$ by adding cuts while column generation expands the current approximation by adding new extreme points. Cutting plane uses primal information to generate new valid inequalities to improve approximation to conv$(\mathcal{S})$ while column generation uses dual information to improve the approximation to conv$(\mathcal{S})$. Thus, the two methods have a primal dual correspondence, and are equivalent, i.e., cutting plane in primal space is equivalent to column generation in dual space.

1.2.4 Disjunction of polytopes

Consider a disjunctive feasible set $\mathcal{S} = \cup_{i=1}^{K} \mathcal{P}_k$ where $\mathcal{P}_k = \{A_k x \leq b_k, 0 \leq x \leq u_k\}$ is a polytope in $\mathbb{R}^d$. In other words, the feasible solutions lie in the union of two or more

polytopes. This can be modeled using binary variable $z_k \in \{0,1\}$ indicating whether x is in the k^{th} polytope, and creating k copies of variables $x \in \mathbb{R}^d$ as follows.

$$\begin{aligned}
\sum_{k\in[K]} x_k &= x \\
A_k x_k &\le b_k z_k \quad \forall k \in [K] \\
0 \le x_k &\le u_k z_k \quad \forall k \in [K] \\
\sum_{k\in[K]} z_k &= 1 \\
z &\in \{0,1\}^K
\end{aligned}$$

It is known that LP relaxation of the above formulation is the convex hull of the feasible set $\mathcal{S}$ [6].

1.2.5 Binary quadratic program (BQP)

Consider a binary quadratic program of the following kind:

$$\begin{aligned}
&\min \sum_i \sum_j c_{ij} y_i y_j + d_o^\top y \qquad \text{(BQP)} \\
&\sum_i \sum_j a_{ij} y_i y_j + d_c^\top y = b \quad \forall c \in \mathcal{C} \\
&y \in \{0,1\}^n
\end{aligned}$$

Here $\mathcal{C}$ denotes the set of constraints and d_o and d_c are $n-$dimensional vectors. Non-convexity in BQP arises from integrality restrictions as well as bilinear terms $y_i y_j$. McCormick envelopes are used to do an exact linearization of the BQP as we discuss next [84, 77].

McCormick relaxation We denote the McCormick envelope of $(x, y) \in [0, 1]$ as the polytope

$$MC(x, y, z) = \{(x, y, z) \in [0, 1]^3 : z \ge x + y - 1, z \le x, z \le y\}.$$

If $x, y \in \{0, 1\}$, then $MC(x, y, z)$ only contains the point $xy \in \{0, 1\}$ corresponding to the product of x and y.

1.2.6 Linear algebra

Definition 1.11 (Field)**.** A field is a set $\mathbb{F}$ with two binary operations called *addition* and *multiplication* defined on it. The addition and multiplication of two elements $a, b \in \mathbb{F}$ will be denoted by $a + b$ and ab respectively, and since binary operations are defined on $\mathbb{F}$, $a + b, ab$ are in $\mathbb{F}$ for any $a, b \in \mathbb{F}$. A field $\mathbb{F}$ must satisfy the following axioms:

a) $a + b = b + a$

b) $(a + b) + c = a + (b + c)$

c) $ab = ba$

d) $a(bc) = (ab)c$

e) $a(b + c) = ab + ac$

f) $\mathbb{F}$ contains additive identity 0 such that $a + 0 = a$ and multiplicative identity 1 distinct from 0 such that $1a = a\ \forall a \in \mathbb{F}$.

g) For all $a \in \mathbb{F}$, there is $-a \in \mathbb{F}$ such that $a + (-a) = 0$.

h) For all $a \neq 0$ in $\mathbb{F}$, there is an element a^{-1} such that $aa^{-1} = 1$.

Definition 1.12 (Vector space)**.** Let $\mathbb{F}$ be a field and V a set. Consider a binary operation on V called *addition* which assigns to each pair of elements u and v of V, a unique sum $u + v \in V$. Consider a second operation, called *scalar multiplication*, which assigns to any $k \in \mathbb{F}$ and any $u \in V$, a unique scalar multiple $ku \in V$. The set V with the binary operations addition and scalar multiplication is a vector space if the following axioms are satisfied:

a) $u + v = v + u$

b) $u + (v + w) = (u + v) + w$

c) There is an additive identity $0 \in V$ such that $u + 0 = u$ for all $u \in V$.

d) For all $u \in V$ there is $-u \in V$ such that $u + (-u) = 0$.

e) $1u = u$

f) For all $k, l \in \mathbb{F}$ and $u \in V$, $(kl)u = k(l(u))$.

g) For all $k \in \mathbb{F}$, $k(u + v) = ku + kv$.

h) For all $k, l \in \mathbb{F}$ and $u \in V$, $(k + l)u = ku + lu$.

Definition 1.13 (Linearly independent)**.** A set of vectors $\{v_1, v_2, \ldots, v_n\}$ from a vector space V is said to be linearly independent if the vector equation

$$a_1v_1 + a_2v_2 + \cdots + a_nv_n = 0$$

has only the trivial solution $a_1 = a_2 = \cdots = a_n = 0$.

Definition 1.14 (Basis)**.** A *basis* $B = \{v_1, v_2, \ldots, v_n\}$ of a vector space V over a field $\mathbb{F}$ is a linearly independent subset of V that spans V, i.e., for every vector $v \in V$, one can choose $a_1, a_2, \ldots, a_n \in \mathbb{F}$ such that $v = a_1v_2 + a_2v_2 + \cdots + a_nv_n$.

Definition 1.15 (Subspace)**.** A subset W of a vector space V is called a subspace of V if W is itself a vector space under the addition and scalar multiplication defined on V. The dimension of a nonzero subspace H is the number of vectors in any basis for H.

Definition 1.16 (Subspaces associated with matrices)**.** Let $A \in \mathbb{F}^{d\times n}$ where $\mathbb{F}$ is a field.

- The *column space* of A, denoted by $\mathcal{C}(A)$, is the span of the columns of A. In other words, we treat the columns of A as vectors in $\mathbb{F}^d$ and take all possible linear combinations of these vectors to form the span. So $\mathcal{C}(A)$ is a subspace of $\mathbb{F}^d$.
- The *row space* of A, denoted by $\mathcal{R}(A)$ is given by $\mathcal{C}(A^T)$. So $\mathcal{R}(A)$, is a subspace of $\mathbb{F}^n$.

Definition 1.17 (Rank)**.** Given a $d \times n$ matrix X, rank of X is the maximum number of linearly independent column vectors in X and is denoted as rank $(X) = r$.

Theorem 1.18. *The rank of a matrix X is the dimension of its row and column spaces.*

In this work, we use dimension of the subspace and rank of the corresponding basis matrix interchangeably.

Definition 1.19 (Galois Field)**.** A Galois Field or a finite field is a field in which there exists finitely many elements. The Galois field is denoted as $\mathbb{F}_p^n$ where p is called characteristic of the field (p is always a prime number) and p^n is the order of the field. For example, $\mathbb{F}_2^3$ contains 8 elements:

$$\mathbb{F}_2^3 = (001, 010, 011, 100, 101, 110, 111)$$

Finite fields of order 2^n are called binary fields and follow modulo-2 arithmetics which coincides with logical XOR for addition and logical AND for multiplication , i.e., $1 \oplus 1 = 0, 1 \oplus 0 = 1, 0 \oplus 0 = 0$ and $0 \otimes 1 = 0, 0 \otimes 0 = 0, 1 \otimes 1 = 1$ We use $\oplus$ to denote addition over $\mathbb{F}_2^n$ and $\otimes$ to denote multiplication over $\mathbb{F}_2^n$. We denote Galois Field of characteristic 2 as $\mathbb{F}_2$.

1.3 Contribution and roadmap

In Chapter 2, we study the problem of scheduling drilling and fracturing of wells in the development of an unconventional oil field. A key challenge in scheduling these operations is the presence of *conflicts* between different operations. A conflict refers to the restriction

that when a pad (collection of wells) is being fractured, it is not allowed to perform drilling or production on any pads within a specified neighborhood of that pad. We propose a new MILP formulation for determining a schedule for drilling and fracturing pads in an unconventional oil field which considers capacity, operational, precedence, and interference constraints. We also propose a formulation that uses more decision variables, but which provides a stronger linear programming relaxation. Due to the large problem size, solving the full MILP model for instances with many pads and a large number of time periods is intractable. Thus, we also derive a MILP-based rolling horizon framework that solves a sequence of limited horizon, coarser-scale MILP instances in a rolling forward fashion to obtain a solution to the full horizon problem on the daily time scale.

In Chapter 3, we study an integer programming approach for subspace clustering with missing data. *Subspace clustering with missing data* (SCMD) is the task of identifying clusters of vectors belonging to the same subspace in a partially observed data matrix whose columns are assumed to lie in a union of K subspaces. We propose a novel mixed-integer linear programming (MILP) solution framework for this 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.

In Chapter 4, we again focus on matrix completion problem but now in the binary setting. We consider low-rank matrix factorization and completion over $\mathbb{F}_2$. Given a binary matrix X observed on indices Ω, we consider the rank r factorization model where the goal is to find binary matrices U and V such that the error on observed entries $\sum_{(ij)\in\Omega} |X_{ij} - Z_{ij}|$ is minimized while ensuring $Z = U \otimes V$. We first derive formulations making use of McCormick envelopes for the product of two binary variables: a base formulation using a general integer variable and an extended formulation using ideas from disjunctive programming and parity polytopes. The latter formulation characterizes the convex hull of the dot product of two vectors in an extended space. We then derive a novel formulation based on a new class of valid inequalities that also characterizes the convex hull of the dot product of two vectors in the original space of variables. Furthermore, we derive new classes of valid inequalities linking dot products of two matrix elements.

2 MIXED-INTEGER LINEAR PROGRAMMING FOR SCHEDULING UNCONVENTIONAL OIL FIELD DEVELOPMENT

2.1 Introduction

We propose a novel MILP formulation for determining a schedule for drilling and fracturing partial pads in an unconventional oil field. Our main contribution is a MILP-based rolling horizon framework that solves a sequence of limited horizon, coarser-scale MILP instances in a rolling forward fashion to obtain a solution to the full horizon problem on the daily time scale.

Unconvention oil field development

Shale oil, a type of unconventional oil, is light crude oil contained in petroleum-bearing formations of low permeability sandstone. The defining characteristic is that the rock is not sufficiently permeable to allow oil to flow out from merely drilling a hole into the formation. However, creating fissures in the rock by injecting water (along with sand and some other chemicals) at high pressure through the formation helps oil to seep back through the cracks and be extracted [41].

The Green River Formation, which covers parts of Colorado, Utah, and Wyoming, has the largest known oil shale deposits in the world, holding about 4.285 trillion barrels of oil. With a threshold of 15 gallons per ton of shale, the prospective oil shale represents a 165-year supply of oil for the United States. This estimate is with respect to the present U.S. demand for petroleum products which is about 20 million barrels per day [13]. Shale oil production in the U.S. has risen from 0.5 million barrels per day to 6.5 million barrels per day between 2008 and 2018. The U.S. Energy Information Administration (EIA) estimates that in 2018, production from shale oil resources accounted to about 61% of total U.S. crude oil production [115]. The rapid rise of shale oil production can be attributed to innovative advancements in horizontal drilling, hydraulic fracturing, and other well stimulation technologies [90]. These advanced production technologies involved in shale oil extraction require high level of investments and thus the strategic development of an oil field is important for the profitability of such fields.

The life cycle of a typical shale oil well starts with drilling the vertical part of the well which is followed by horizontal drilling. The next step in the well development process is hydraulic fracturing (fracturing, for short) which consists of pumping a mixture of water, sand, and chemicals to stimulate the rock to allow oil and gas to escape through the rock.

After fracturing is completed, a cleaning crew cleans out and turns the well online, after which the well begins producing oil.

The field can be represented by a collection of partial pads P as shown in Figure 2.1. A partial pad $p \in P$ is a piece of land containing a number of wells. In this work, following common practice, we assume all wells in a partial pad are first drilled sequentially (by a single drilling crew) and then fractured sequentially (by a single fracturing crew). Thus, for scheduling purposes we treat all the wells on a partial pad p as a single entity. We use the term *partial pad* in this chapter to refer to any number of wells that should be drilled sequentially and then fractured sequentially. In this way, we differ from the conventional definition of a *pad* — a temporary drilling site — which can hold several dozen wells. Clearly, not all wells of such large pads (sometimes referred to as "mega pads") should be drilled or fractured sequentially. However, when development planners lay out large pads, they typically arrange the wells in rows that are drilled sequentially. The main reason why sequential drilling and fracturing is done is efficiency gains: for drilling, these include "skidding" or "walking" of the rig; for fracturing, these include "zipper-frac" (shortening drilling and fracturing durations, respectively). Additionally, if well operations are carried out sequentially, rig mobilization and demobilization costs are reduced. In practice, development planners therefore typically do not consider drilling or fracturing wells individually, but in groups — which, in this paper, we refer to as *partial pads* [118]. Since mobilization costs are controlled by the policy of drilling and fracturing all wells within a partial pad together, we do not otherwise consider the mobilization and demobilization costs of the drilling rigs/fracturing crews trips to and from the partial pads in our model.

Literature survey

A key challenge in scheduling operations in unconventional oil field development is the presence of *conflicts* between different operations. A conflict refers to the restriction that when a partial pad is being fractured, it is not allowed to perform drilling or production on any partial pads within a specified neighborhood of that partial pad. There are very few papers in the literature that consider such conflicts. An important exception is the work of Ondeck et al. [91], who provide an optimization framework for selecting and developing gas wells on a single partial pad. Their model considers conflicts between wells within the partial pad in addition to other detailed considerations such as the possibility to curtail gas production and the expenses involved in mobilizing development resources. In contrast, we focus on the scheduling of development operations at the field level, using a model that incorporates fewer details of the individual well development, but which can

be used to schedule development activities of multiple drilling and fracturing crews over the development of the entire unconventional field. Our model is targeted to be used by development planners, so we assume that strategic decisions concerning how many wells to be drilled and fractured in a single visit of a crew have been made *a priori.*. We employ a partial pad-level aggregation and integrate the model into a rolling horizon framework.

There is significant literature on conventional oil and gas infrastructure planning and development, e.g., [111, 59, 78, 50, 27]. The literature on unconventional oil and gas planning and development is more limited. In recent years, some work has been done to determine an optimal structure of a shale gas network. Cafaro and Grossmann [22] determined the most profitable supply chain design by using a branch-refine-optimize (BRO) strategy to solve a mixed-integer nonlinear programming (MINLP) formulation. Knudsen and Foss [61] proposed a formulation to solve the scheduling of multi-well shut-ins. Arredondo-Ramirez et al. [4] proposed a method for determining a superstructure with potential wells, gas treatment plants, and distribution networks. Drouven and Grossmann [36] also presented a superstructure capturing the tree structure of gas gathering systems. They solved a nonconvex MINLP to consider spatial gas quality variations within multiple delivery node gathering systems. Another important aspect of field development is scheduling of different operations. Iyer et al. [59] discussed a discrete-time MILP model for conventional offshore oil field infrastructure development and used a decomposition approach to solve larger instances. Cafaro et al. [21] and Drouven and Grossmann [35] introduced optimization frameworks to plan shale gas well refracture treatments of a single well under uncertainty. Rahmanifard and Plaksina [99] optimized the well placement in a shale gas reservoir and compared the performance of different heuristics for maximizing well production.

The scheduling problem is related to the flexible flow shop scheduling problem, which has been proven to be NP-Hard [49, 52, 121]. Flexible flow shop scheduling has been intensively studied in many industries [74]. Gupta [52] considered a two-stage scheduling problem in which a set of jobs is given, each of which has to undergo two processes in sequence. There are a set of identical resources available to do each of the two processes. The well development scheduling problem has a similar structure, where the wells are the jobs, the first process is drilling, the second process is fracturing, and the resources are the drilling and fracturing crews. Our model extends the flexible flow shop scheduling model by considering the conflicts between these processes (i.e., it is not allowed to drill or produce while fracturing a nearby well). Because of this additional important complication, methods for flexible flow shop scheduling cannot be directly applied to our problem.

The rolling horizon approach has been applied in a variety of applications where a

MILP is solved over a smaller number of periods in successive iterations. Some recent work using the rolling horizon approach in various applications are [83, 107, 105, 109]. A unique feature of our rolling horizon strategy is that in addition to solving a sequence of problems with a limited lookahead, the problems we solve have a coarser time-scale than the time-scale of the solution we produce.

Contributions

We propose a novel MILP formulation for determining a schedule for drilling and fracturing partial pads in an unconventional oil field which considers capacity, operational, precedence, and interference constraints. We also propose a formulation that uses more decision variables, but which provides a stronger linear programming relaxation. Our results show that this larger formulation can improve solution times by 25-70% on instances with relatively few time periods in the planning horizon, but is not advantageous for instances with more time periods. Due to the large problem size, solving the full MILP model for instances with many partial pads and a large number of time periods is intractable. Thus, we also derive a MILP-based rolling horizon framework that solves a sequence of limited horizon, coarser-scale MILP instances in a rolling forward fashion to obtain a solution to the full horizon problem on the daily time scale. We benchmark this approach against a baseline scheduling algorithm that approximates current practice, where partial pads are scheduled in the order of their discounted production profit with limited lookahead to avoid conflicts. Our results show that our proposed MILP-based rolling horizon approach can improve net present value of a field by 4-6%. Considering that large fields such as in the Permean basin are estimated to have billions of barrels of oil [42, 43], improvements in this range can be very economically significant. While we focus on oil field development, our results may also be useful for planning shale gas field development, which follows a similar development process.

This chapter is organized as follows. We provide a detailed description of the problem, our MILP formulation, and its use within a rolling horizon framework in Section 2.2. We provide the alternative, larger MILP formulation in Section 2.2.4, and also discuss how the strength of the formulation can be obtained without adding additional variables by using a cutting plane algorithm. In Section 2.4, we present results of a computational study in which we compare the different MILP formulations, study the effects of period length and lookahead window in the rolling horizon approach, and quantify the value of our MILP-based rolling horizon approach by benchmarking against a baseline scheduling algorithm.

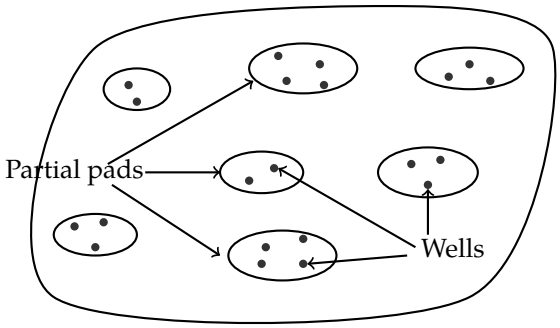


Figure 2.1: Diagrammatic view of partial pads and wells in a shale oil field.

2.2 Problem description and solution approach

2.2.1 Problem statement

We consider an oil field that has a collection of partial pads P to be developed as shown in Figure 2.1. A partial pad $p \in P$ is a piece of land containing a number of wells. The first step in developing a partial pad $p \in P$ is drilling, which takes $\hat{\tau}_p^d$ days. The second step is fracturing which takes $\hat{\tau}_p^f$ days. Note that we use $(\hat{})$ to denote the time durations of parameters in days. For example, $\hat{\tau}_p^d$ denotes drilling in days whereas in section 2.2.3 we will use τ_p^d to represent the approximate number of *periods* to drill partial pad p for a given coarser time discretization. Drilling and fracturing operations have a fixed cost c_p^d and c_p^f associated with them, which are assumed to be charged at the beginning of the operation. An important assumption of our model is that we know how many wells are to be operated upon in a single visit by a crew. The collection of wells which are to be processed in a single crew visit is treated as a partial pad in the model. The final steps in the development of a partial pad are cleaning and turning in line operations, but as these are typically done after fracturing without delay and they cause no conflicts with other operations, we do not consider them in our model.

A fixed number of drilling (n^d) and fracturing (n^f) crews are available, so that at any point in time at most n^d partial pads can be in the process of drilling and at most n^f partial pads can be in the process of fracturing. At most one drilling crew or fracturing crew can be assigned to a partial pad. Moreover, each partial pad p is drilled and fractured in a single visit of drilling and fracturing crew respectively, without interruption. A partial pad p starts producing oil after fracturing is complete. The amount of oil produced from a partial pad in a period depends on the amount of time since the partial pad began fracturing, and is specified by a production curve $\hat{\alpha}_p$, where $\hat{\alpha}_{pk}$ represents the amount of oil production from partial pad p during day k after fracturing was started. Since production cannot occur

while a partial pad is being fractured and cleaned the production curve is zero for the initial periods until fracturing and cleaning are complete, then increases to the partial pad's actual initial production level, and typically decreases over time after that. The production curve of a partial pad is obtained by summing the production curves of the individual wells on the partial pad.

For two partial pads within some distance threshold, fracturing one partial pad and drilling/producing the other cannot occur concurrently, for one of the following reasons:

- For simultaneous fracturing and *drilling*, the main reasons are either *operational* — a rig drilling one row of wells is physically blocking another row of wells to be fractured — or concerning *Safety, Health and Environment* (SH&E).
- For simultaneous fracturing and *producing*, the main reasons are so-called parent-child fracture interferences. Operators have observed fracture interferences between existing production wells (parent wells) and newly fractured wells (child wells), and these typically have a negative impact on the production from both the parent and child wells [82].

For each partial pad $p \in P$ the set $N_p \subseteq P$ represents the neighboring partial pads for which production and drilling are prohibited when partial pad p is being fractured. Although it is not necessary for our model, we assume the neighborhoods are symmetric so that $p \in N_q$ if and only if $q \in N_p$ for $p, q \in P$. If a partial pad that has completed fracturing is shut down due to fracturing at a neighboring well, we assume that the production profile of the partial pad still progresses to the next time period. Since the production rate curves are decreasing, when production resumes it will be at a lower rate. We let P^{oil} denote the net revenue (price less processing costs) per barrel of oil and assume it is known and fixed for the entire time horizon of the field development process. The problem is to determine the drilling and fracturing start time of each partial pad in the field in order to maximize the net present value (NPV) of net revenues obtained from the field over its production horizon, where NPV is calculated using an annual discount rate of i^A. We let $i^D = \sqrt[365]{i^A + 1} - 1$ denote the equivalent daily discount rate.

Illustrative example

We consider a field consisting of 20 partial pads with three drilling crews and one fracturing crew. The partial pads are distributed on a rectangular grid with three rows and seven columns as shown in Figure 2.2. In this example, the neighbors of a partial pad p consist of the pads lying immediate adjacent to p horizontally or vertically. The drilling duration

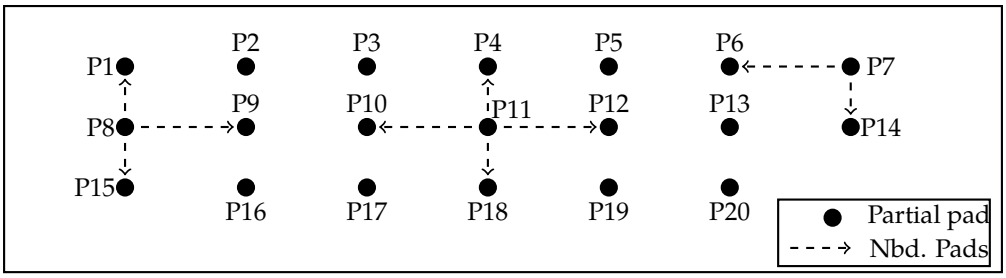


Figure 2.2: Layout of partial pads in the illustrative example. The neighbors of partial pads P8, P11, and P7 are illustrated with dashed arrows.

of partial pads is between 40 and 210 days and the fracturing duration is between 14 and 42 days. A sample schedule for this instance is shown in Figure 2.3. Day 0 in this schedule is set as Jan 1, 2019. The schedule demonstrated here is generated using a baseline scheduling algorithm (Algorithm 2) which is discussed in Section 2.4.3. We see that on every day, there are no more than three drilling and one fracturing operations happening in parallel. The order of operations on each partial pad is drilling, followed by fracturing, and then production. We also observe that fracturing a partial pad leads to a shut down in the production of neighboring partial pads (shown as gaps in the production bars). For instance, production on P2 is halted when P1 is fractured. Another observation is that sometimes a crew may need to idle due to interference constraints. For instance, a drilling crew becomes free after drilling at P2 is completed. However, drilling at P1 isn't initiated right away. P2 is first fractured after which drilling at P1 begins. This is because P1 and P2 lie in the same neighborhood and hence they cannot be drilled and fractured simultaneously. Schedules which have higher NPV tend to limit the number of production shutdowns due to fracturing conflicts, limit idling of resources, and begin production of high volume wells earlier.

2.2.2 Coarse-time discretization

We formulate the pad drilling and fracturing scheduling problem as a MILP problem using a discrete-time model consisting of a set of time periods $T := \{0, 1, \ldots, |T|\}$. In order to obtain a more compact model, we assume a period consists of D days. For a pad p which takes $\hat{\tau}_p^d$ days to drill, we approximate its drilling time in periods, τ_p^d, by rounding $\hat{\tau}_p^d/D$ to the nearest integer. Similarly, the fracturing duration in periods, τ_p^f, is approximated by rounding $\hat{\tau}_p^f/D$ to the nearest integer. The parameter D provides a trade-off between model accuracy and complexity. A larger value of D leads to a problem with a shorter time horizon which is hence more compact, but also leads to more inaccuracy due to rounding.

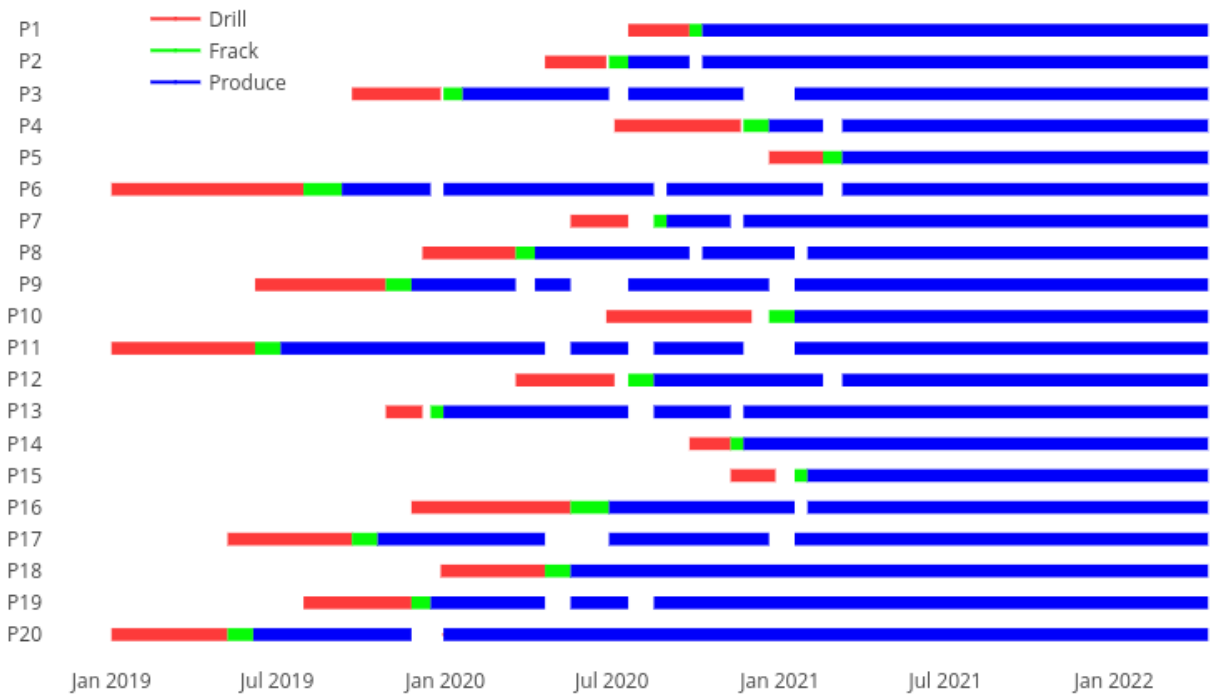


Figure 2.3: Sample schedule for the illustrative example, obtained from the baseline scheduling algorithm. The NPV of this schedule is \$5.499*10^9.

Given the annual discount rate i^A, the periodic discount rate i is given by the formula

$$i = \sqrt[N]{i^A + 1} - 1,$$

where N represent the number of periods in a year.

For a pad $p \in P$, the amount of oil produced in the t^{th} period after fracturing was complete is computed as

$$\alpha_{pt} = \sum_{k=tD}^{(t+1)D-1} \hat{\alpha}_{pk}. \tag{2.1}$$

If a pad begins production during the planning horizon, then our model needs to account for all production of the pad from the end of the planning horizon until the pad no longer produces. To do so, for each pad $p \in P$ and $0 \leq t \leq |T| - \tau_p^f$, we define β_{pt} to be the discounted total revenue from oil produced beyond the planning horizon if fracturing of pad p begins at time period t, where the revenue is discounted to period $|T|$. Specifically, β_{pt} is computed as

$$\beta_{pt} = P^{oil} \sum_{k=|T|-(t+\tau_p^f)}^{\text{TMAX}_p} (1+i)^{-k} \alpha_{pk}. \tag{2.2}$$

The oil that is produced *within* the planning horizon is accounted for differently because of the possibility of production shut downs due to fracturing operations at neighboring pads, and this is why β_{pt} must be calculated separately for each possible fracturing start time t.

The notation used in our problem definition is summarized in Table 2.1.

Parameter	Description	**Units**
P	Set of pads	-
N_p	Neighboring pads of p	-
D	Length of a period	days
$\hat{\tau}^d$	Drilling duration	days
τ^d	Rounded drilling duration	Periods
$\hat{\tau}^f$	Fracturing duration	days
τ^f	Rounded fracturing duration	periods
c^d	Drilling cost	\$
c^f	Fracturing cost	\$
n^d	Number of drilling crews	-
n^f	Number of fracturing crews	-
$\hat{\alpha}_{pk}$	Pad production on k^{th} day since fracturing began	barrels
α_{pt}	Pad production on t^{th} period since fracturing began	barrels
i^A	Annual discount rate	1/year
i	Periodic discount rate	1/period
i^D	Daily discount rate	1/day
P^{oil}	Net revenue from a barrel of oil	\$/barrel

Table 2.1: A summary of parameters used in Chapter 2

2.2.3 MILP formulation

The decision variables in the MILP model are as follows:

- x_{pt} : Binary variable that takes the value 1 if drilling at pad $p \in P$ starts at the beginning of time period $t \in T$, 0 otherwise.
- y_{pt}: Binary variable that takes the value 1 if fracturing at pad $p \in P$ starts at the beginning of time period $t \in T$, 0 otherwise.
- $\overline{x}_{pt}$: Binary variable that takes the value 1 if drilling for pad $p \in P$ has been completed before the beginning of time period $t \in T$, 0 otherwise.
- $\overline{y}_{pt}$: Binary variable that takes the value 1 if fracturing for pad $p \in P$ has been completed before the beginning of time period $t \in T$, 0 otherwise.
- w_{pt}: Binary variable that takes the value 1 if pad $p \in P$ is in production mode in period $t \in T$, 0 otherwise.

- v_{pt}: Amount of oil produced from pad $p \in P$ during period $t \in T$.

The objective is to maximize net present value (NPV) of net revenue:

$$NPV = \sum_{t \in T} \sum_{p \in P} \Big[(1+i)^{-t} \big(P^{oil} v_{pt} - c_p^d x_{pt} - c_p^f y_{pt} \big) + (1+i)^{-t} \beta_{pt} y_{pt} \Big]. \tag{2.3}$$

Note that the first term is discounted using the period discount rate i since the expression $P^{oil} v_{pt} - c_p^d x_{pt} - c_p^f y_{pt}$ represents the net revenue in period t. The term $\beta_{pt} y_{pt}$ is discounted t periods since the computation of β_{pt} discounts the revenues of oil from beyond the planning horizon to time period t.

Next, we introduce the constraints in the model.

Relationship Constraints: The first set of constraints relates the decision variables for determining when drilling starts for a pad to the decision variables that indicate whether or not drilling has been completed:

$$\overline{x}_{pt} = 0, \qquad \forall p \in P, \quad t = 0, 1, \ldots, \tau_p^d - 1, \tag{2.4a}$$

$$\overline{x}_{pt} = \overline{x}_{p,t-1} + x_{p,t-\tau_p^d}, \qquad \forall p \in P,\ t = \tau_p^d, \tau_p^d + 1, .., |T|. \tag{2.4b}$$

Equations (2.4a) record the fact that for each pad $p \in P$ it is not possible to have completed drilling within the first $\tau_p^d - 1$ periods. Equations (2.4b) are equivalent to the equations

$$\overline{x}_{pt} = \sum_{k=0}^{t-\tau_p^d} x_{pk}, \quad \forall p \in P,\ t = \tau_p^d, \tau_p^d + 1, \ldots, |T|, \tag{2.5}$$

and thus correctly capture the relationship that drilling at a pad p is complete if and only if drilling was started at time period $t - \tau_p^d$ or earlier. Note that we use (2.4b) in our formulation rather than (2.5) because the number of constraints is the same, and the set of constraints (2.4b) has significantly fewer nonzero coefficients. The constraints (2.4b) also imply the equation $\sum_{t=0}^{|T|} x_{pt} = \overline{x}_{p|T|} \leq 1$, which thus enforces the condition that each pad is drilled at most once. Observe that the model allows a pad p to not be selected for drilling at all (i.e., $\overline{x}_{p|T|} = 0$). This is necessary because, as we discuss in Section 2.3, for large-scale instances the formulation will be used within a rolling horizon framework in which the problem is solved over a limited lookahead horizon. Due to the limited length of the lookahead horizon it may not be feasible to drill all the wells within the horizon.

A similar set of constraints relates the decision variables for determining when fracturing starts for a pad to the decision variables that indicate whether or not fracturing has been

completed:

$$\overline{y}_{pt} = 0, \qquad \forall p \in P, \quad t = 0, 1, \ldots, \tau_p^f - 1, \tag{2.6a}$$

$$\overline{y}_{pt} = \overline{y}_{p,t-1} + y_{p,t-\tau_p^f}, \qquad \forall p \in P, \quad t = \tau_p^f, \tau_p^f + 1, .., |T|. \tag{2.6b}$$

Capacity Constraints: The following constraints ensure that the number of pads being simultaneously drilled or fractured doesn't exceed the number of drilling or fracturing crews available at any period [63]:

$$\sum_{p \in P} \sum_{k=(t-\tau_p^d+1)_+}^{t} x_{pk} \leq n^d, \qquad \forall t \in T, \tag{2.7}$$

$$\sum_{p \in P} \sum_{k=(t-\tau_p^f+1)_+}^{t} y_{pk} \leq n^f, \qquad \forall t \in T. \tag{2.8}$$

Here we use the notation $(z)_+ = \max\{0, z\}$ for any integer z. Note that a pad $p \in P$ is being drilled at time t if drilling has begun in one of the periods τ_p^d before t, thus the expression on the left-hand side of (2.7) computes the number of pads being drilled at time t, and similarly for fracturing in (2.8).

Precedence Constraints: We next consider constraints that enforce that drilling must be done before fracturing. Specifically, the following constraints ensure that if drilling for a pad $p \in P$ has not yet been completed by a time t, then fracturing cannot be completed by time $t + \tau_p^f$:

$$\overline{y}_{p,t+\tau_p^f} \leq \overline{x}_{pt}, \quad \forall p \in P,\ t = 0, 1, \ldots, |T| - \tau_p^f. \tag{2.9}$$

Similarly, production can occur only after a pad has completed fracturing. The following constraint therefore enforces that if fracturing is not yet complete on a pad $p \in P$ by time period t, then time period t cannot be a production period:

$$w_{pt} \leq \overline{y}_{pt}, \qquad \forall p \in P,\ t \in T. \tag{2.10}$$

Vicinity Constraints: For each pad p and time period t, if any pad $q \in N_p$ is being fractured at time t, then p cannot be in the process of drilling during that period, nor can it be producing during that period:

$$w_{pt} + \sum_{k=(t-\tau_p^d+1)_+}^{t} x_{pk} \leq 1 - \sum_{k=(t-\tau_q^f+1)_+}^{t} y_{qk}, \qquad \forall p \in P,\ q \in N_p,\ t \in T. \tag{2.11}$$

Oil Production Constraints: The volume of oil that can be produced from pad p in time period t (v_{pt}) is bounded above based on the production curve and when fracturing of the pad began:

$$v_{pt} \leq \sum_{k=0}^{t-\tau_p^f} \alpha_{p,t-k} y_{pk}, \qquad \forall p \in P, t = \tau_p^f, \tau_p^f + 1, \ldots, |T|. \tag{2.12}$$

If $y_{pk} = 0$ for all $k \leq t - \tau_p^f$, then fracturing is not yet complete by time t and hence (2.12) correctly records that no production can occur in period t. Otherwise, if $y_{pk} = 1$ for some $k \leq t - \tau_p^f$, then (2.12) bounds the production to not exceed $\alpha_{p,t-k}$, which is the limit in period t since in this case fracturing began $t - k$ periods before period t.

In addition, the production amount from a pad must be zero if the pad is shut down due to fracturing at a neighboring pad. A shut down of pad p in time period t due to fracturing in a neighboring pad will cause $w_{pt} = 0$ due to constraints (2.11). Thus the following constraint then ensures that the volume produced from pad p in period t is zero if shut down occurs:

$$v_{pt} \leq \bar{\alpha}_p w_{pt}, \qquad \forall p \in P, \quad t \in T, \tag{2.13}$$

where $\bar{\alpha}_p$ is an upper bound on the maximum possible production from pad p in a period (e.g., $\bar{\alpha}_p = \max\{\alpha_{pk} : k \geq 0\}$).

Note that when $w_{pt} = 1$ for a pad $p \in P$ in a time period $t \in T$, the production amount v_{pt} will be exactly equal to the expression in the right-hand side of (2.12) due to the objective function. We must use inequality in the constraint (2.12) in order to allow $v_{pt} = 0$ in the case that $w_{pt} = 0$.

For our model, the inequalities (2.12) and (2.13) imply that after the shut-down of a pad, production resumes at the level it would have been if the shutdown had never occurred. This assumption is motivated by computational experience with an earlier version of this model. Specifically, in an earlier version of this work, we approximated the production curve $\hat{\alpha}_p$ as a piecewise linear function. Using the binary variables that represent the breakpoints of the piecewise linear function, we could model the impact of shutdowns in more detail, and we could introduce discount factors that represented the level at which a well resumes production after a shutdown. Agreeing with the findings of Crafton and Noe [32], our data indicated that shutdowns are generally harmful and consequently should be avoided in early well/pad production where the production curve is steep. The results of our optimization runs showed the same, as good solutions avoided shutdowns shortly after starting pad operations. Comparing the higher-fidelity models with the one presented here, we noted that the computational cost of the higher-fidelity model could not be justified,

as we would typically obtain the same optimal solution using the lower-fidelity model presented in this chapter. We therefore decided to model production using the assumption that the production rate resumes at the level it would have been if shutdown did not occur.

Strengthening constraints: Since drilling must be done before fracturing, and a pad cannot be producing until fracturing is complete we add the following constraints that prohibits a pad from having more than one operation (drilling, fracturing, or producing) occuring at any time period t:

$$w_{pt} + \sum_{k=t-\tau_p^d+1}^{t} x_{pk} + \sum_{k=t-\tau_p^f+1}^{t} y_{pk} \leq 1, \qquad \forall p \in P,\ t = \tau_p^d, \tau_p^d + 1, ..., |T|. \tag{2.14}$$

Note that for binary feasible solutions (2.14) is implied by the constraints (2.5), (2.6), (2.9), and (2.10). However, (2.14) is not implied for fractional solutions, and we find that adding it provides minor improvement in the linear programming relaxation of the formulation.

In summary, the MILP formulation is to maximize the objective (2.3), subject to the constraints (2.4), (2.6) - (2.14), and with binary restrictions on the decision variables $x_{pt}, y_{pt}, \overline{x}_{pt}, \overline{y}_{pt}, w_{pt}$ for $p \in P, t \in T$, and non-negativity on the oil production variables, $v_{pt} \geq 0$ for $p \in P, t \in T$. We refer to this formulation as MILP1.

2.2.4 Alternative MILP formulation

In this section we present an alternative MILP formulation and demonstrate that the LP relaxation of this formulation is at least as strong as the LP relaxation of MILP1 presented in Section 2.2.3. We refer to this alternative formulation as MILP-EF, as it can be considered to be an extended formulation since it uses more decision variables.

For each pad $p \in P$, we introduce a new set of binary decision variable z_{pkt} for $\tau_p^f \leq t \leq |T|$ and for $k \leq t - \tau_p^f$, where $z_{pkt} = 1$ if pad p begins fracturing in period k *and* is producing in period t ($w_{pt} = 1$), and $z_{pkt} = 0$ otherwise.

The first set of new constraints in the MILP-EF formulation relates the new z_{pkt} variables to the w_{pt} variables, enforcing the logic that if $w_{pt} = 0$ for a period t, then all the z_{pkt} variables for $k \leq t - \tau_p^f$ must also be zero:

$$\sum_{k=0}^{t-\tau_p^f} z_{pkt} \leq w_{pt}, \qquad \forall p \in P,\ t = \tau_p^f, \ldots, |T|. \tag{2.15}$$

This enforces the desired logic when $w_{pt} = 0$, and when $w_{pt} = 1$, this simply states the

redundant constraint that fracturing for pad p can be started at most once in periods 0 to $t - \tau_p^f$.

The next relationship constraint we add enforces the condition that if a pad p does not have fracturing start in a period k ($y_{pk} = 0$), then all of the z_{pkt} variables for $t \geq k + \tau_p^f$ must be zero:

$$z_{pkt} \leq y_{pk}, \qquad \forall p \in P,\ k = 0, \ldots, t - \tau_p^f,\ t = \tau_p^f, \ldots, |T|. \tag{2.16}$$

Finally, we can replace the constraints that specify the upper bounds of volume produced in each period (inequalities (2.12) and (2.13)) with the following constraints:

$$\sum_{k=0}^{t-\tau_p^f} \alpha_{p,t-k} z_{pkt} = v_{pt}, \qquad \forall p \in P, t \in T, \tag{2.17}$$

where by convention for $t < \tau_p^f$ the left-hand side sum is zero and thus for such t the constraints simply enforce the condition that no production can occur until after fracturing is complete. For $t \geq \tau_p^f$, the production amount in period t is set to zero when $w_{pt} = 0$, since in this case $z_{pkt} = 0$ for all $k \leq t$ by (2.15). On the other hand, when $w_{pt} = 1$, the variables z_{pkt} will be equal to y_{pk} for all $k \leq t - \tau_p^f$ in an optimal solution since this is allowed by (2.16) and because the objective is improved by increasing v_{pt}. Thus, if fracturing begins in some period $k' \leq t - \tau_p^f$, then we will have $z_{pk't} = y_{pk'} = 1$, and so the production amount in period t will be set to $v_{pt} = \alpha_{p,t-k'}$, which is the correct amount given that fracturing began k' periods earlier.

In summary, the new formulation MILP-EF maximizes the objective (2.3), subject to the constraints (2.4), (2.6) - (2.11), (2.14), and (2.15) -(2.17).

Lemma 2.1. *The LP relaxation upper bound of MILP-EF is not larger than the LP relaxation upper bound of formulation MILP1.*

Proof. We show that if $(x, \overline{x}, y, \overline{y}, v, w, z)$ is a feasible solution to the LP relaxation of MILP-EF, then $(x, \overline{x}, y, \overline{y}, v, w)$ is a feasible solution to the LP relaxation of MILP1. This proves the claim since the objective functions in the two models are the same.

Thus, let $(x, \overline{x}, y, \overline{y}, v, w, z)$ be a feasible solution to the LP relaxation of MILP-EF. We only need to verify that this solution satisfies (2.12) and (2.13), since all other constraints in MILP1 are included in MILP-EF. First, for each $p \in P$ and $t \geq \tau_p^f$ we obtain

$$v_{pt} = \sum_{k=0}^{t-\tau_p^f} \alpha_{p,t-k} z_{pkt} \leq \sum_{k=0}^{t-\tau_p^f} \alpha_{p,t-k} y_{pk}$$

where the equality follows from (2.17) and the inequality follows from (2.16), and hence the solution satisfies (2.12).

Next, for each $p \in P$ and $t \geq \tau_p^f$ we obtain

$$v_{pt} = \sum_{k=0}^{t-\tau_p^f} \alpha_{p,t-k} z_{pkt} \leq \sum_{k=0}^{t-\tau_p^f} \bar{\alpha}_p z_{pkt} \leq \bar{\alpha}_p y_{pt}$$

where the first equality follows from (2.17), the first inequality follows because $\alpha_{p,t-k} \leq \bar{\alpha}_p$, and the second inequality follows from (2.15). For $t < \tau_p^f$ (2.17) implies $v_{pt} = 0$. Thus, in either case this verifies that the solution satisfies (2.13). □ □

2.2.5 Valid inequalities in the original variable space

The number of decision variables and constraints in MILP-EF grows quadratically with the number of time periods $|T|$, and thus for problems with many time periods it may be time-consuming to even solve the LP relaxation of this model. We thus discuss how the strength of this formulation can be obtained in the space of variables of the original model MILP1 by adding valid inequalities as cuts to the LP relaxation. One possible implementation of this would be to add these valid inequalities at the initial LP relaxation before starting the branch-and-bound process for solving MILP1.

Cut separating linear program. Given a (partial) solution $(\hat{y}, \hat{v}, \hat{w})$ of the LP relaxation of MILP1, we can determine if there is a solution to the LP relaxation of formulation MILP-EF by solving a small linear program for each $p \in P$ to determine if there are values for the variables z_{tkp} that satisfy constraints (2.15) -(2.17) for this pad p. Specifically, for each pad $p \in P$ we solve the linear program:

$$\min \sum_{k \in T} \gamma_t \tag{2.18a}$$

$$\text{s.t.} \sum_{k=0}^{t-\tau_p^f} \alpha_{p,t-k} z_{tk} + \gamma_t = \hat{v}_{pt}, \quad \forall t \in T, \tag{2.18b}$$

$$\sum_{k=0}^{t-\tau_p^f} z_{tk} \leq \hat{w}_{pt}, \qquad \forall t = \tau_p^f, \ldots, |T|, \tag{2.18c}$$

$$0 \leq z_{tk} \leq \hat{y}_{pt}, \qquad \forall k = 0, \ldots, t - \tau_p^f,\ t = \tau_p^f, \ldots, |T|, \tag{2.18d}$$

$$\gamma_t \geq 0, \qquad \forall t \in T. \tag{2.18e}$$

Given $\hat{w}_{pt}, \hat{y}_{pt}$, and $\hat{v}_{pt} \geq 0$, (2.18) is feasible because one can set all z_{tk} variables equal zero and $\gamma_t = \hat{v}_{pt}$ for all $t \in T$. This LP is also trivially bounded, and hence it has an optimal solution. The optimal value of (2.18) is zero if and only if there exist values of z_{ptk} for $k = 0, \ldots, t - \tau_p^f,\ t = \tau_p^f, \ldots, |T|$ that satisfy the pad p constraints of MILP-EF. Thus, such an extended solution for pad p exists if and only if the dual objective value of every feasible dual solution of (2.18) is less than or equal to zero.

Let π, ρ, and θ be the dual variables associated with constraints (2.18b), (2.18c), and (2.18d), respectively, and let $\hat{\pi}$, $\hat{\rho}$, $\hat{\theta}$ be an optimal dual solution. Observe that the dual objective is to maximize

$$\sum_{t \in T} \pi_t \hat{v}_{pt} + \sum_{t=\tau_p^f}^{|T|} \rho_t \hat{w}_{pt} + \sum_{t=\tau_p^f}^{|T|} \hat{y}_{pt} \sum_{k=0}^{t-\tau_p^f} \theta_{tk}.$$

Thus, if the optimal value of (2.18) is zero for every $p \in P$, then there exist values of the z_{ptk} variables such that appending these values to the partial solution $(\hat{y}, \hat{v}, \hat{w})$ of the LP relaxation of MILP1 is feasible to MILP-EF. On the other hand, if the optimal value of (2.18) is positive for some $p \in P$, the following inequality is implied by the constraints of MILP-EF, and hence by correctness of that formulation, is valid for formulation MILP-1, i.e., it does not cut off any integer feasible solutions:

$$\sum_{t \in T} \hat{\pi}_t v_{pt} + \sum_{t=\tau_p^f}^{|T|} \hat{\rho}_t w_{pt} + \sum_{t=\tau_p^f}^{|T|} y_{pt} \sum_{k=0}^{t-\tau_p^f} \hat{\theta}_{tk} \leq 0. \tag{2.19}$$

Moreover, this inequality is violated by the current LP relaxation solution $(\hat{y}, \hat{v}, \hat{w})$. Thus, adding this inequality to the LP relaxation of MILP1 and then re-solving has the potential to improve the LP relaxation value. This process can be repeated in a simple cutting plane algorithm in which in each iteration LP relaxation of MILP1 with cuts added is solved. Using the LP relaxed solution of MILP1, the cut separating linear programs (2.18) are solved for each $p \in P$, and then cuts of the form (2.19) are added to MILP1 when violated. If the process continues until no more violated cuts are found, the resulting LP relaxation value will be equal to the LP relaxation value of MILP-EF.

Cut validation. Validity of the inequality (2.19) requires that the dual solution $(\hat{\pi}, \hat{\rho}, \hat{\theta})$ used to construct it be a feasible dual solution. When solving a linear program in practice, the solution given may be slightly infeasible (within numerical tolerances). Using such a solution has the potential to lead to an invalid cut. To ensure validity of the cut (2.19), we

propose to modify the (potentially infeasible) solution returned by the LP solver to make it feasible as follows.

The feasible region to the dual linear program of (2.18) is given by the inequalities

$$\alpha_{p,t-k}\pi_t + \rho_t + \theta_{tk} \leq 0, \qquad \forall k = 0, \ldots, t - \tau_p^f,\ t = \tau_p^f, \ldots, |T|, \tag{2.20a}$$

$$\pi_t \leq 1, \qquad \forall t \in T, \tag{2.20b}$$

$$\rho_t \leq 0, \qquad \forall t = \tau_p^f, \ldots, |T|, \tag{2.20c}$$

$$\theta_{kt} \leq 0, \qquad \forall k = 0, \ldots, t - \tau_p^f,\ t = \tau_p^f, \ldots, |T|. \tag{2.20d}$$

Given an "approximately" feasible dual solution $(\hat{\pi}, \hat{\rho}, \hat{\theta})$, we propose to adjust it to a guaranteed feasible solution $(\overline{\pi}, \overline{\rho}, \overline{\theta})$ using the following formulae:

$$\begin{aligned}
\overline{\pi}_t &= \min\{\hat{\pi}_t, 1\}, \quad \forall t \in T, \\
\overline{\rho}_t &= \min\{\hat{\rho}_t, 0\}, \quad \forall t = \tau_p^f, \ldots, |T|, \\
\overline{\theta}_{tk} &= \min\{0, -(\alpha_{p,t-k}\overline{\pi}_t + \overline{\rho}_t)\}, \qquad \forall k = 0, \ldots, t - \tau_p^f,\ t = \tau_p^f, \ldots, |T|.
\end{aligned}$$

2.3 Rolling horizon implementation

We next present a rolling horizon approach which is designed to obtain solutions for the problem for significantly larger instances. The basic idea with a rolling horizon framework is to solve a model over a limited planning horizon, fix the initial decisions, then move the window of the planning horizon forward in time and repeat. We let $\zeta = |T|D$ be the number of days in the planning horizon of the optimization model. In addition to limiting the planning horizon, we also use time periods of length D days to limit the size of the MILP formulation being solved at each step. However, the rolling forward is done at the daily level, so that in the end the algorithm produces a schedule that is feasible to the problem using a daily time discretization. Figure 2.4 illustrates the basic idea of the approach.

The details of the MILP-based rolling horizon approach are given in Algorithm 1. The status of each pad is maintained throughout the algorithm, which is initialized as 'idle'. The status of a pad is updated to 'drilling' when it is in the process of drilling, and changes to 'drilled' after drilling is complete. When fracturing begins the status is updated to 'fracturing' and finally it is updated to 'fractured' when that is complete, after which all operations for the pad are done. The number of available drilling and fracturing crews at the current day is updated in the variables A^d and A^f, respectively. At the beginning of processing each day, we first check to see if there are any free drilling crews and pads that

need to be drilled, or there are any free fracturing crews and pads that need to be fractured. If so, the limited horizon, aggregate time-period MILP model is solved (line 6). In this MILP, the decision variables for any drilling or fracturing operations currently in progress are fixed to require them to begin in the initial time period, and their durations are adjusted according to the remaining duration of these operations. After solving the MILP, the pads that are assigned to begin drilling in the first period of the model horizon are stored in M_t^{drill}, and likewise the pads that are assigned to begin fracturing in the first period of the MILP model are stored in M_t^{fracture}. Then, for each pad $p \in M_t^{\text{drill}}$ we first check to see if starting drilling on pad p is feasible with respect to conflicts between pads currently being fractured in its neighborhood, and if it is feasible we update the pad status, store its start day in drillstart[p], and update the number of available drilling crews (line 12). The conflict check is necessary because the MILP formulation uses aggregate time periods, and so could potentially miss conflicts when creating the schedule at the daily basis. A similar process is performed for assigning fracturing operations for each pad $p \in M_t^{\text{fracture}}$. At the end of processing each day, we determine whether any pads with status of 'drilling' or 'fracturing' will complete that process at the end of the day, and if so, update their status and the number of drilling or fracturing crews available. For pads that have status 'drilling' or 'fracturing' and which are not completing that operation on that day, we update the remaining time of these operations in terms of the number of periods. We ensure that drilling or fracturing duration is at least one period to prevent rounding down the duration to zero periods. Finally, we check the termination condition of the algorithm, which occurs when all pads have status 'fractured', indicating that all drilling and fracturing operations have been scheduled.

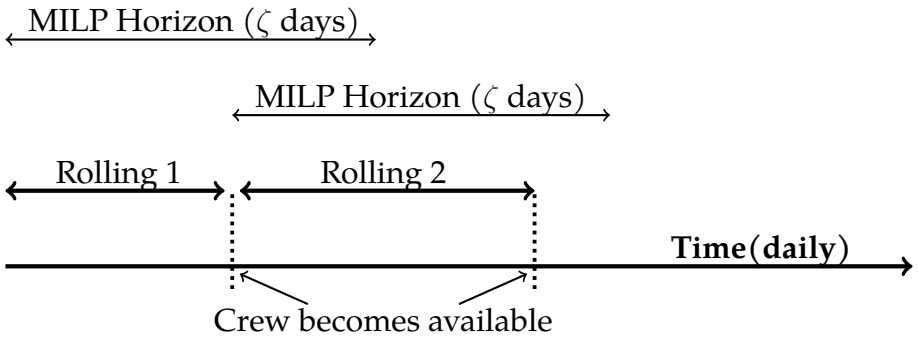


Figure 2.4: Illustration of the rolling horizon approach. The limited horizon MILP model is re-solved whenever a crew becomes free, which is checked at the daily level.

Algorithm 1: MILP based rolling horizon algorithm

status$[p]\leftarrow$ 'idle' for all $p \in P$;
$A^d \leftarrow n^d$, $A^f \leftarrow n^f$;
for $t = 0, 1, 2, \ldots$ **do**
  Determine if there is possibility to assign an operation to start in time t;
  if ($A^d > 0$ *and* $\exists\ p$ *s.t. status*$[p]$='*idle*') *or* ($A^f > 0$ *and* $\exists\ p$ *s.t. status*$[p]$='*drilled*')
    Solve the MILP model for next $|T|$ periods (ζ days) ;
    $M_t^{\text{drill}} \leftarrow$ set of pads $p \in P$ which MILP solution assigns to drill in period 0;
    $M_t^{\text{fracture}} \leftarrow$ set of pads $p \in P$ which MILP solution assigns to fracture in period 0;
    for $p \in M_t^{drill}$ **do**
      Check for interference;
      if $\nexists q \in N_p$ *s.t. status*$[q]$='*fracturing*' *and* $A^d > 0$
        drillstart$[p]\leftarrow$ t, $A^d \leftarrow A^d - 1$, status$[p]\leftarrow$'drilling' ;
    for $p \in M_t^{fracture}$ **do**
      Check for interference;
      if $\nexists q \in N_p$ *s.t. status*$[q]$='*drilling*' *and* $A^f > 0$
        frac start$[p]\leftarrow t$, $A^f \leftarrow A^f - 1$, status$[p]\leftarrow$'fracturing';
  Update the completed operations and remaining duration for the pads under operation;
  for $p \in P$ *s.t. status*$[p]$='*drilling*' **do**
    if $t =$*drillstart*$[p]+\hat{\tau}_p^d$
      $A^d \leftarrow A^d + 1$, status$[p]\leftarrow$'drilled';
    else
      $\tau_p^d \leftarrow \max\Big(\text{round}\big(\frac{\hat{\tau}_p^d - (t - \text{drillstart}[p])}{D}\big), 1\Big)$;
  for $p \in P$ *s.t. status*$[p]$='*fracturing*' **do**
    if $t =$*fracstart*$[p]+\hat{\tau}_p^f$
      $A^f \leftarrow A^f + 1$, status$[p]\leftarrow$'fractured';
    else
      $\tau_p^f \leftarrow \max\Big(\text{round}\big(\frac{\hat{\tau}_p^f - (t - \text{fracstart}[p])}{D}\big), 1\Big)$;
  Exit if all pads have been fractured and ready to produce;
  if *status*$[p]$='*fractured*' *for all* $p \in P$
    break;

2.4 Computational study

We next report results from a computational study in which we compare the performance of the two proposed MILP formulations, MILP1 and MILP-EF, investigate the effect of the period length and lookahead window parameters of the MILP-based rolling horizon approach, and compare the solutions obtained with the proposed MILP-based rolling horizon approach to those obtained by a baseline scheduling algorithm that mimics current practice.

2.4.1 Test instances

We randomly generated test instances for our experiments using parameter ranges adopted from [21], [36] , [23], and [91]. Each instance consists of a set of pads which we assume lie on a regular grid of length L and width W. We define pads to be neighbors of each other if they are immediately adjacent vertically or horizontally in the grid. The data for each pad is determined by first randomly choosing how many wells are on the pad, which we generate uniformly as an integer between one and six. We then randomly generate characteristics of each well on the pad and determine the pad data based on the wells. The characteristics of the wells are generated according to the distributions given in Table 2.2. For a pad $p \in P$, this yields a set of wells $\mathcal{W}_p$ that are on that pad. We then determine the pad parameters from the wells on the pad as follows: $\hat{\tau}_p^d = \sum_{w \in \mathcal{W}_p} \hat{\tau}_w^d$, $\hat{\tau}_p^f = \sum_{w \in \mathcal{W}_p} \hat{\tau}_w^f$, $\mathrm{TMAX}_p = \max\{\mathrm{TMAX}_w : w \in \mathcal{W}_p\}$, $c_p^d = \sum_{w \in \mathcal{W}_p} c_w^d$, $c_p^f = \sum_{w \in \mathcal{W}_p} c_w^f$. For each well w, we follow [3] and model the oil production rate function of the well using an exponential decline curve

$$\lambda_w(s) = M_w e^{-a_w s}, \tag{2.21}$$

where $s \geq 0$ represents the time (in days) from when production begins. The daily production curve parameters for each pad p and day $t \geq 0$ are then computed as

$$\alpha_{pt} = \begin{cases} 0 & t < \hat{\tau}_p^f \\ \sum_{w \in \mathcal{W}_p} \int_{t-\hat{\tau}_p^f}^{t-\hat{\tau}_p^f+1} \lambda_w(s) ds & t \geq \hat{\tau}_p^f. \end{cases}$$

We use an annual discount rate of $i^A = 0.12$ and set $P^{oil} = \$60$ per barrel.

2.4.2 Comparison of MILP formulations

We first investigate the performance of formulations MILP1 and MILP-EF proposed in Sections 2.2.3 and 2.2.4, respectively. For each instance, we compare the computational

Parameter	Description	Value	Units
$\hat{\tau}_w^d$	Time to drill the well	Normal, $\mu = 30, \sigma = 10$	days
TMAX_w	Life of a well	40	years
$\hat{\tau}_w^f$	Time to fracture the well	Normal, $\mu = 7, \sigma = 2$	days
c_w^d	Cost to drill the well	$1.5 + .003\hat{\tau}_p^d$	MM$
c_w^f	Cost to fracture the well	$3.5 + .01\hat{\tau}_p^d$	MM$
M_w	Initial production rate of well	Uniform$(400, 3300)$	Barrels/day
a_w	Decay rate constant for well	Uniform(.0003-0.0007)	1/day

Table 2.2: Well-specific instance parameters.

Inst.	Pads	ζ (Periods)	ζ (Days)	Period len.	$\mathbf{n^d, n^f}$
I-10-67	10	67	1000	15 days	3,1
I-10-71	10	71	500	7 days	5,2
I-15-20	15	20	600	30 days	3,1
I-20-20	20	20	900	45 days	3,1

Table 2.3: Characteristics of instances used to compare MILP1 and MILP-EF.

time and nodes explored to reach optimality or a desired optimality gap. We used Gurobi 8.1.1 as the MILP solver. Additionally we set a time limit of 15000s (250 min) for these experiments. These experiments were performed on a 2.8 GHz Quad-Core Intel Core i7 with 16 GB RAM.

The characteristics of the test instances used in this study are presented in Table 2.3. For each instance size (row in Table 2.3) we generated five random instances with those characteristics. We solve instances I-10-67, I-10-71, and I-15-20 to the gap of 0.01%. For the larger instance I-20-20 we solve to optimality gap 1%.

We report the gap of LP relaxation from the optimal value (LP gap), solution time, and nodes explored in the tree to reach the specified optimality gap for the two formulations on each test instance in Table 2.4. We can see that on an average the number of nodes explored to reach a solution of desired optimal tolerance is fewer by a factor in the range of 1.2-3.6 when using MILP-EF as compared to MILP1. This is a consequence of the better LP relaxation of MILP-EF, which is also demonstrated by the smaller LP relaxation relaxation gap. When considering solution time, we observe that although we explore fewer nodes with MILP-EF in instances I-10-67 and I-10-71, the solution times using MILP1 are significantly smaller. On the other hand, the solution times are smaller for MILP-EF on instances I-15-20 and I-20-20. This also includes an instance I-20-20-C where MILP1 reached the time limit of 15000s while formulation MILP-EF was solved to the 1% optimality gap in 3805 seconds. The difference in solution time behavior can be explained by the size of the model, and in particular the number of periods. Instances I-10-67 and I-10-71 have 67

and 71 periods, respectively, and hence the formulation MILP-EF, which has size growing quadratically in the number of periods, gets very large. We conclude that for instances with a small number of time periods (e.g., ≤ 20) MILP-EF formulation is preferred, but for instances with more time periods formulation MILP1 may be preferred.

Our primary interest in this work is to demonstrate the impact of the use of the MILP formulations within the rolling horizon framework for generating high-quality solutions on larger instances, which we investigate in the next two subsections. Thus, we did not computationally test the approach of using cutting planes to obtain the strength of formulation MILP-EF in the space of the formulation MILP1.

Inst.	**Opt. sol**	**LP gap(%)**		**IP time (sec.)**		**# Nodes**	
	x(10^9)	MILP1	MILP-EF	MILP1	MILP-EF	MILP1	MILP-EF
I-10-67-A	2.841	3.72	2.66	571	708	11069	5112
I-10-67-B	3.309	3.98	2.94	415	980	8720	6039
I-10-67-C	3.536	3.35	2.43	416	548	6133	2295
I-10-67-D	3.240	4.60	3.32	1894	1704	39220	10508
I-10-67-E	2.727	3.26	2.21	157	393	2692	1909
Avg.		3.78	2.71	690.6	866.6	13566.8	5172.6
I-10-71-A	2.311	4.23	2.79	496	760	11446	6391
I-10-71-B	2.694	3.45	2.50	214	418	7910	4432
I-10-71-C	2.400	3.87	2.56	459	1152	12028	8497
I-10-71-D	2.591	3.37	2.39	433	1465	7706	9522
I-10-71-E	2.527	8.52	7.18	321	1234	10248	9771
Avg.		4.69	3.48	384.6	1005.8	9867.6	7722.6
I-15-20-A	3.517	3.61	2.63	135	83	18460	7244
I-15-20-B	3.338	3.16	1.66	138	84	22370	9774
I-15-20-C	4.189	2.86	1.99	57	38	15956	4795
I-15-20-D	3.804	4.38	3.23	164	87	32282	12616
I-15-20-E	4.179	3.72	2.47	119	144	29538	30647
Avg.		3.55	2.40	122.6	87.2	23721.2	13015.7
I-20-20-A	5.255	6.14	3.87	2294	742	276358	54933
I-20-20-B	5.501	5.40	3.12	326	101	39010	8472
I-20-20-C	6.042	6.24	4.16	15000	3805	1511503	346964
I-20-20-D	5.697	5.91	3.41	566	207	65531	20499
I-20-20-E	5.194	5.89	3.57	2395	1782	205097	158152
Avg.		5.92	3.63	4116.2	1327.4	419499.8	117804

Table 2.4: Comparison of formulations MILP1 and MILP-EF.

2.4.3 Baseline scheduling algorithm

We next turn our attention to the use of the MILP-based rolling horizon approach for generating solutions to instances that are too large to solve to optimality. To provide context for the quality of the solutions generated on these instances, we present the baseline scheduling algorithm, which to the best of our knowledge closely mimics current scheduling practice. The idea behind the baseline scheduling algorithm is to develop pads with the highest discounted revenue first, in order to obtain the revenue from the highest value pads earlier, which is beneficial due to the discounting used in the NPV calculation. Thus, we rank the pads on the basis of their discounted total revenue, which is computed in (2.2) as $\beta_{p,|T|-\tau_p^f}$. The operations are then scheduled by prioritizing the pads with highest discounted volume production first, while ensuring that we don't violate any precedence, capacity, operational and conflict constraints.

The details of the baseline scheduling algorithm are given in Algorithm 2. We create a drill-queue of the pads, ordered highest to lowest by discounted revenue. Pads are added to the fracture-queue (line 20) after their drilling operation is initiated. Note that the order in the two queues may be different as drilling operations may not always start in the preferred order. This happens when there are delays arising from the conflicts i.e. drilling initiation on the pad next in queue can be delayed if a neighborhood pad is being fractured. In the algorithm each pad is initialized with the 'idle' status. We update the status of each pad as it goes through different stages of the development cycle, i.e., {'idle', 'drilling', 'drilled', 'fracturing', 'fractured' }. The variables A^d and A^f keep track of the number of free drilling and fracturing crews available at each point in time. A^d and A^f are initialized with n^d and n^f as all crews are free at $t = 0$. Similarly, the variables $\mu^d[p]$ and $\mu^f[p]$ are used to keep track of the remaining drilling and fracturing duration of each pad p. These are initialized with the actual drilling and fracturing duration ($\hat{\tau}^d[p]$ and $\hat{\tau}^f[p]$). Algorithm 2 proceeds by considering each day in the planning horizon in sequence. For each day, we first assign fracturing operations (lines 6-14) and then drilling operations (lines 15-24) to start that day. Fracturing is prioritized over drilling in case there is a conflict between either starting drilling or fracturing on two neighboring pads. Fracturing is prioritized because once it is complete the pad can begin production and revenue is generated. Fracturing operations are initiated in preference order of the fracture-queue. Fracturing is initiated at the next pad in the queue if there is a free fracturing crew, drilling is complete, and initiating fracturing on the given pad doesn't violate any interference constraints (lines 11-13). If all these conditions hold true, fracturing is started and we store the fracturing start day for the pad in 'fracstart', reduce the number of fracturing crews available (A^f) by one, remove the pad from the fracture queue, and update the status of pad to 'fracturing'

(line 12). If a pad is not assigned for fracturing, but a fracturing crew is still available, the next pad in the queue is considered, and so on until either the full queue has been checked or there are no available fracturing crews. Drilling operations are initiated in a similar way (lines 15-24). However, when initiating drilling operations we check additional conditions in order to avoid creating conflicts with upcoming fracturing operations (line 20). In particular, we check each neighboring pad q to see if it is currently being fractured (status[p]='fracturing') or is ready to be fractured(status[q]='drilled'). We also check for the possibility of fracturing on pad q getting delayed because of initating drilling on pad p in the neighborhood in current period i.e. pad q, currently being drilled, may begin fracturing before the drilling operation on pad p under consideration would be completed To perform this check, we define the values 'earlieststart[q]' for $q \in P$ as follows:

$$\text{earlieststart}[q] = \begin{cases} 0, & \text{if status}[q] = \text{'fracturing' or status}[q] = \text{'drilled'}, \\ +\infty, & \text{if status}[p] = \text{'idle' or status}[q] = \text{'fractured'}, \\ \sum_{j=1}^{i} \mu^f[q_j]/n^f, & \text{if } q = q_i \in \text{fracture-queue} = [q_1, q_2, \ldots, q_k]. \end{cases}$$

At the end of processing each day, we determine whether any pads with status of 'drilling' or 'fracturing' will complete that process at the end of the day, and if so update their status and the number of drilling and fracturing crews available. For pads with status 'drilling' or 'fracturing' and which are not completing that day, we update the remaining time of these operations (lines 25-39).

2.4.4 Parameter study for MILP-based rolling horizon approach

We next study the effect of the period length (D) and lookahead horizon (ζ) parameters in the MILP-based rolling horizon framework. Intuitively, one would expect to obtain the best solutions using a very large lookahead horizon ζ and small period length (daily). However, since the MILP becomes larger and more difficult to solve as the number of periods increases, it is necessary to impose a time limit when solving the MILP, and thus use the best solution found within the time limit. If the number of periods is too large, then the solution obtained within the time limit may be significantly suboptimal (or maybe even no solution could be found) leading to poor performance. Thus, we conduct a study to determine values of D and ζ that lead to the best solutions.

The number of periods used in the instances in this study is relatively large (usually more than 20), and hence following the guidelines in Section 2.4.2 we use formulation MILP1 to solve the instances in this study. We set an optimality gap tolerance of 2% and a

Algorithm 2: Baseline Scheduling Algorithm

drill-queue$\leftarrow$ Ordered list of pads based on $\beta_{p|T|}$; fracture-queue=[ ] ;
status[p]$\leftarrow$ 'idle' for all $p \in P$;
$A^d \leftarrow n^d$, $A^f \leftarrow n^f$;
$\mu^d[p] \leftarrow \hat{\tau}^d[p]$, $\mu^f[p] \leftarrow \hat{\tau}^f[p]$ for all $p \in P$;
for *t=0,1,2...* **do**
    Assign fracturing operation;
    for p *in fracture-queue* **do**
        if $A^f = 0$
            break;
        if $\nexists q \in N_p$ *s.t. status*[q]=*'drilling'*
            fracstart[p] $\leftarrow t$, $A^f \leftarrow A^f - 1$, fracture-queue.remove(p), status[p]$\leftarrow$ 'fracturing';
    Assign drilling operation;
    for p *in drill-queue* **do**
        if $A^d = 0$
            break;
        if $\nexists q \in N_p$ *s.t. earliestfrac*[q]$< \hat{\tau}^d_p$
            drillstart[p] $\leftarrow t$, $A^d \leftarrow A^d - 1$, drill-queue.remove(p), status[p]$\leftarrow$'drilling';
            fracture-queue.append(p);
    Update the completed operations and remaining duration for the pads under operation;
    for p *in* P *s.t. status*[p]=*'drilling'* **do**
        if *drillstart*[p]+$\hat{\tau}^d_p = t$
            $A^d \leftarrow A^d + 1$, status[p]$\leftarrow$'drilled';
        else
            $\mu^d[p] \leftarrow \mu^d[p] - 1$;
    for p *in* P *s.t. status*[p]=*'fracturing'* **do**
        if *fracstart*[p]+$\hat{\tau}^f_p = t$
            $A^f \leftarrow A^f + 1$, status[p]$\leftarrow$'fractured';
        else
            $\mu^f[p] \leftarrow \mu^f[p] - 1$;

time limit of 1000s for each solution of the limited-horizon MILP model and use the best solution obtained by the solver when it terminates. As before, we used Gurobi 8.1.1 as the MILP solver. These experiments were obtained on a machine having two 3.2 GHz Intel Xeon CPUs with 4GB RAM. We set the number of threads in Gurobi to four.

For this study, we consider two different instances of moderate size given in Tables 2.5 and 2.6. When varying the lookahead horizon parameter ζ, we consider different possibilities based on the fraction of the time it would take to develop all the pads in the instance. We refer to our estimate of this time as γ, which is calculated as:

$$\gamma = 1.3 * [\frac{\sum_{p \in P} \hat{\tau}_p^d}{n^d} + \frac{\sum_{p \in P} \hat{\tau}_p^f}{n^f}].$$

Here, the factor 1.3 is used to account for possible idling of drilling or fracturing crews due to conflict constraints. Note that, although instance I55 (Table 2.6) has only slightly more pads to be developed than I45 (Table 2.5), the estimated total planning horizon γ is significantly longer because in I55 (Table 2.6) there are fewer drilling and fracturing crews.

The results of these experiments are reported in Tables 2.7 and 2.8. In our case study we use $D = 15, 30,$ and 45 days for period length and use lookahead windows of 20%, 40%, 60%, and 80% of the entire planning horizon (γ). For each combination of period length (D) and lookahead horizon (ζ), we report the percent improvement in NPV of the rolling horizon solution over the solution obtained by the baseline scheduling algorithm (negative percentage indicates the solution was worse than that obtained from the baseline scheduling algorithm). We also report the fraction of the limited-horizon MILP instances for which the time limit was reached before the desired optimality gap of 2% was reached (Time-lim. frac.) and the average ending optimality gap of the limited-horizon MILP instances (Mean opt. gap).

Parameter	**Value**
Pads	45
Crews(Drill, Fracture)	6,2
Length of Horizon (γ)	1350 days

Table 2.5: Instance I45 description.

Parameter	**Value**
Pads	55
Crews(Drill, Fracture)	3,1
Length of Horizon (γ)	2700 days

Table 2.6: Instance I55 description.

Period len. (D)	**Lookahead horizon**(ζ)	$\lvert T\rvert$	**Imprv.**(%)	**Time-lim. frac.**	**Mean opt. gap**(%)
15	0.2 γ	18	3.35	0.00	1.17
15	0.4 γ	36	4.04	0.34	1.80
15	0.6 γ	54	3.59	0.36	2.63
15	0.8 γ	72	3.73	0.39	2.25
30	0.2 γ	9	3.13	0.00	1.05
30	0.4 γ	18	3.41	0.00	1.59
30	0.6 γ	27	3.77	0.28	1.81
30	0.8 γ	36	4.04	0.27	1.85
45	0.2 γ	6	-1.80	0.00	1.18
45	0.4 γ	12	1.14	0.00	1.65
45	0.6 γ	18	1.16	0.38	2.14
45	0.8 γ	24	1.80	0.42	2.28

Table 2.7: Results of MILP-based rolling horizon algorithm on instance I45. NPV of baseline scheduling algorithm solution=\$$1.4293 * 10^{10}$.

Period len. (D)	**Lookahead horizon** (ζ)	$\lvert T\rvert$	**Imprv.**(%)	**Time-lim. frac.**	**Mean opt. gap**(%)
15	0.2 γ	36	5.21	0.00	1.48
15	0.4 γ	72	5.30	0.16	1.79
15	0.6 γ	108	5.24	0.45	2.29
15	0.8 γ	144	5.19	0.44	18.13
30	0.2 γ	18	2.78	0.00	1.47
30	0.4 γ	36	4.97	0.03	1.68
30	0.6 γ	54	5.10	0.30	1.81
30	0.8 γ	72	5.22	0.29	1.95
45	0.2 γ	12	2.61	0.00	0.90
45	0.4 γ	24	3.56	0.00	1.63
45	0.6 γ	36	3.15	0.34	1.89
45	0.8 γ	48	3.54	0.37	2.20

Table 2.8: Results of MILP-based rolling horizon algorithm on instance I55. NPV of baseline scheduling algorithm solution=\$$1.7721 * 10^{10}$.

The results in Tables 2.7 and 2.8 confirm that using a longer lookahead horizon usually yields better solutions, and that this holds for each period length D. On the other hand, while there is generally a significant improvement when lookahead is increased from 0.2γ to 0.4γ, there is little to no improvement obtained by increasing beyond that. In both instances, for the shortest period length, $D = 15$, we observed that the quality of the rolling horizon

solution was actually smaller for the larger lookahead horizons. This can be explained by looking at the fraction of instances that were terminated due to the time limit and the mean optimality gap of the limited-horizon MILP problems in these instances. In particular, a significant fraction of these problems were terminated due to the time limit, and their average optimality gap was relatively large, indicating that the solutions obtained may have been significantly suboptimal.

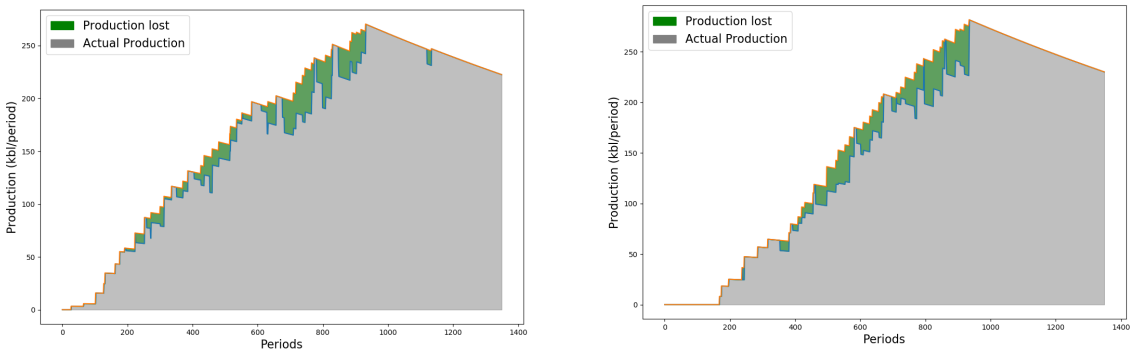
Next we discuss the effect of period length on the quality of the solutions obtained. We observe from Tables 2.7 and 2.8 that when the period length D is 45 days, the quality of the solutions obtained is significantly lower than when using $D = 15$ or $D = 30$. Using $D = 15$ tends to provide the best solutions, although the difference in the solutions obtained using $D = 15$ and $D = 30$ is small provided the lookahead horizon is 0.4γ or larger.

Thus, in these experiments we found that a lookahead window of 0.4γ and period length of either $D = 15$ or $D = 30$ yields the best results overall. We emphasize, however, that because the solution of the MILP problems was stopped after a time limit, the values of these parameters that yield the best solutions are dependent on the time limit used, as well as the MILP solver and computational environment used. In addition, when using the MILP-based rolling horizon algorithm in practice, it is only necessary to solve a single limited-horizon MILP instance whenever a drilling or fracturing crew completes an operation and becomes available. Given that in real time there may be several weeks between such events, a significantly longer time limit could potentially be used for solving these instances, thus enabling use of a longer lookahead window and smaller period length.

To give some insight, we show a visual comparison between performance of the rolling horizon and baseline scheduling approaches using the instance I-45. Figure 2.5 shows the lost production of oil because a pad had to be shutdown during production. It is clear from Figure 2.5 that the amount of production lost when using the rolling horizon approach is less than the amount lost in the solution obtained from the baseline scheduling algorithm. In Figure 2.6 we show periodic and cumulative production for the instance. It is clear from Figure 2.6a that the rolling horizon approach yields higher production in the initial periods compared to the baseline scheduling algorithm. This leads to higher NPV. Moreover, we also observe from Figure 2.6b that the cumulative production curve obtained by the rolling horizon approach is always above that obtained by the baseline scheduling algorithm.

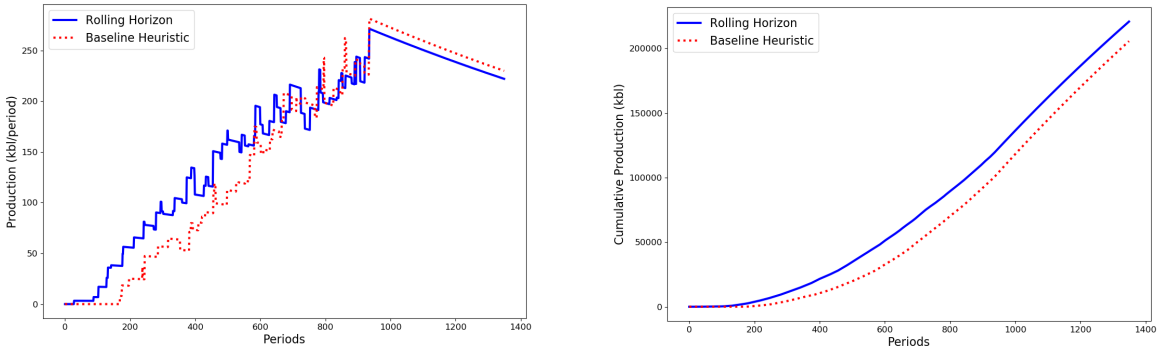
2.4.5 Estimating optimality loss of the rolling horizon approach

Our proposed rolling horizon framework allows solving scheduling instances over time-frames significantly longer than can be solved using a single MILP formulation over the



(a) Rolling Horizon (b) Baseline Scheduling Algorithm

Figure 2.5: Lost production of oil from the field for I45 ($D = 30, |T| = 18$).



(a) Perioidic oil production (b) Cumulative oil production with time

Figure 2.6: Output comparsion of the baseline heuristic with the rolling horizon approach for I45 ($D = 30, |T| = 18$).

whole time horizon. However, this approach is not guaranteed to find an optimal solution. In this section, we attempt to quantify the suboptimality of solutions obtained using the MILP+Rolling Horizon (RH) framework. We do this by comparing the solutions obtained when solved to optimality using the MILP1 formulation against solutions obtained in our RH framework on instances that are small enough to be solved over the full planning horizon. In particular, we consider instances in which the base time period is longer than a day (we use 7 and 15 days), so that when implementing our RH framework, we force the solves to "roll forward" the horizon only in units of full periods, rather than days.

The characteristics of the test instances used in this study are given in Table 2.9. For each instance type, we randomly generate five instances (A,B,C,D,E). The experiments are performed in the same computational environment as Section 2.4.4. For this experiment, we solve instances of MILP1 with no time limit and an optimality gap of 0.01%, and a time limit is set to 1000s for each call to the model in the rolling horizon implementation.

For MILP1, we solve the model with its lookahead window as the full time horizon (γ Periods-MILP1 in Table 2.9), and in MILP+Rolling Horizon we solve the model with a lookahead window of $\zeta = 0.6\gamma$ (Periods-RH in Table 2.9).

Inst.	**Pads**	γ **Periods-MILP1**	ζ **Periods-RH**	**Period len.**	n^d, n^f
I-10-67	10	67	41	15 days	3,1
I-10-71	10	71	43	7 days	5,2
I-14-100	14	100	60	7 days	3,1

Table 2.9: Characteristics of instances used to compare MILP1 and MILP+Rolling Horizon (RH).

The results comparing revenue obtained in the instances used in this study are reported in Table 2.10. As expected, the MILP1 revenue is always higher than the Rolling Horizon(RH) revenue, as MILP1 generates the best possible schedule. However, the difference in quality of solution between the two approaches is modest, averaging 0.14%, 0.70%, and 1.23%, respectively for the three instance types in the study. Thus we conclude that with a sufficiently long lookahead time-horizon, the RH framework can deliver solutions that are nearly optimal.

	MILP1	**RH**($\zeta = 0.6\gamma$)	**Diff (%)**	**RH**($\zeta = 0.4\gamma$)	**Diff (%)**
I-10-67-A	3.1842	3.1840	0.00	3.1840	0.00
I-10-67-B	3.6630	3.6540	0.24	3.6259	1.01
I-10-67-C	3.8951	3.8951	0.00	3.8661	0.74
I-10-67-D	3.6306	3.6302	0.01	3.6116	0.05
I-10-67-E	3.0771	3.0715	0.18	3.0382	1.26
I-10-71-A	2.6967	2.6928	0.14	2.6209	2.81
I-10-71-B	3.1104	3.0876	0.73	2.9382	5.53
I-10-71-C	2.7644	2.7634	0.04	2.7029	2.22
I-10-71-D	2.8204	2.8154	0.18	2.7828	1.33
I-10-71-E	2.9471	2.8753	2.43	2.6217	11.04
I-14-100-A	3.5768	3.5385	1.07	3.4483	3.59
I-14-100-B	4.2950	4.2491	1.06	4.2369	1.35
I-14-100-C	4.9629	4.7934	3.40	4.6991	5.31
I-14-100-D	4.6553	4.6408	0.31	4.2651	8.38
I-14-100-E	3.9600	3.9458	0.35	3.8361	3.12

Table 2.10: Revenue(x10^9) comparison of MILP+Rolling Horizon Framework with MILP1

Instance	Pads	n^d, n^f	γ (days)
I-80-160	80	6,2	2400
I-90-100	90	9,3	1500
I-100-170	100	6,2	2550
I-110-180	110	6,2	2700

Table 2.11: Large-scale instance data.

Instance	Baseline NPV ($)	Roll. Imprv.(%)	Time-lim. frac.	Mean opt. gap(%)
I-80-160	$2.7339 * 10^{10}$	5.13	0.47	2.45
I-90-100	$3.0461 * 10^{10}$	5.01	0.19	1.79
I-100-170	$3.3928 * 10^{10}$	4.47	0.59	2.70
I-110-180	$3.7399 * 10^{10}$	6.54	0.59	8.38

Table 2.12: Improvement using MILP-based rolling horizon framework.

2.4.6 Large-scale instances

We finally test our rolling horizon framework on larger-scale instances using the parameters $D = 15$ and $\zeta = 0.4\gamma$, as suggested in Section 2.4.4. The instances we use for this study are described in Table 2.11. We report the baseline scheduling algorithm NPV and the improvement obtained by the MILP-based rolling horizon algorithm in Table 2.12. As in Tables 2.7 and 2.8, we also report the fraction of MILP instances for which the time limit was reached and the mean optimality gaps of the MILP instances. The results in Table 2.12 demonstrate that the solutions obtained using our MILP-based rolling horizon algorithm consistently have significantly higher NPV than those obtained with the baseline scheduling algorithm. For these large-scale instances, a 5% improvement in performance translates to roughly $1.5B.

2.5 Conclusion and future work

In this chapter we presented a novel MILP-based rolling horizon algorithm to schedule drilling and fracturing operations in an unconventional oil field development while considering the interaction effects between various pads. We provided two MILP formulations for the limited horizon MILP solved as part of this approach. The second formulation provides better LP relaxation bounds, which translates to shorter solution times for instances with a small number of time periods, but for larger instances the first formulation was found to be more effective. A key feature of the rolling horizon approach we propose is that it yields a solution at the daily time-scale, while solving a sequence of coarser time-scale MILP problems. An empirical study demonstrated that the approach can be used to plan

development of fields with more than 100 pads, and the solutions obtained have 4-6% higher NPV than solutions obtained with a baseline scheduling algorithm that mimics current practice.

Our work assumed all data, including the drilling and fracturing durations, are deterministic. In reality, these are estimated via forecasts that may have significant errors. The rolling horizon framework we propose can naturally be applied in this setting, by using the updated state of the system, and updated estimates of the durations, whenever a new limited-horizon MILP is solved. However, an interesting direction for future work is to investigate the use of a stochastic or robust optimization formulation of the scheduling problem within the rolling horizon framework to see if this may yield improved solutions.

3 INTEGER PROGRAMMING APPROACH TO SUBSPACE CLUSTERING WITH MISSING DATA

3.1 Introduction

In this chapter, we propose a novel mixed-integer linear programming (MILP) solution framework for the subspace clustering with missing data problem. Our key contribution is a MILP-based framework which uses column-generation approach for identifying candidate subspaces combined and Benders decomposition approach for solving the linear programming relaxation of the formulation. The proposed framework is also capable of self-determining the number of subspaces and their dimensions.

Subspace clustering with missing data

Given a partially observed data matrix $X_\Omega \in \mathbb{R}^{d\times n}$, where Ω is the set of observed indices of matrix X, *subspace clustering with missing data* (SCMD) is the task of identifying clusters of vectors belonging to the same subspace. Column vectors, $X_1, X_2, \ldots, X_n$ of X 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 $r_i < d$ and there a total of K subspaces.

If the clustering of points is known, then the data matrix can be completed using well-known methods for low-rank matrix completion (LRMC) [26, 25, 24, 7, 102, 96, 89]. The SCMD problem has applications in modern machine learning in many areas such as image classification [71, 124], motion segmentation [112, 101], and recommendation systems [100].

Literature survey

Non MILP methods Subspace clustering was first studied for fully-observed data ($\Omega = \{(ij) : \forall i \in [d], j \in [n]\}$), see [1] for a review of subspace clustering methods with fully-observed data. Most of the methods for subspace clustering with missing data have been extended 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 [94] where the authors show that for ambient dimension d and low dimensional subspaces of rank r, $O(rd)$ columns per subspace are both necessary and sufficient.

The most celebrated and dominant approach in subspace clustering algorithms is based on the self-expressiveness property, originally proposed for fully-observed data by

Elhamifar and Vidal [39]. Self-expressive methods learn a sparse representation of the data by solving an optimization problem of the following kind:

$$C^* = \arg\min \|X - XC\|_F^2 + \lambda\rho(C) \qquad \text{s.t. } \text{diag}(C) = 0. \tag{3.1}$$

Self-expressive methods have been studied extensively for different choices of $\rho(\cdot)$, e.g., ℓ_1, ℓ_2, and nuclear norm [39, 79, 80, 125, 81, 37, 92, 123, 119]. C_{ij} can be interpreted as the link between points i and j. The segmentation is obtained by applying spectral clustering on graph G with adjacent matrix $A = |C| + |C|^T$ which uses k-means on eigenvectors of Laplacian of G [88].

Self-expressive methods for subspace clustering have been extended to the case of missing data. Let Ω_j denotes the set of observed dimensions 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. [122] proposed to zero-fill the missing entries in X to get X_{ZF} and then solve (3.1) while restricting the loss to observed entries, i.e.,

$$X_{ZF} = \begin{cases} X_{ij}, & \text{if } (i,j) \in \Omega, \text{i.e., } X_{ij} \text{ is observed} \\ 0, & \text{if } (i,j) \notin \Omega, \text{i.e., } X_{ij} \text{ is not observed} \end{cases} \tag{3.2}$$

$$C^* = \arg\min \|I_\Omega \circ (X_{ZF} - X_{ZF}C)\|_F^2 + \lambda\rho(C) \qquad \text{s.t. } \text{diag}(C) = 0. \tag{3.3}$$

Tsakiris and Vidal [113] and Charles et al. [29] studied the theoretical conditions under which solution to (3.3) is subspace preserving, i.e., each data point is only connected to points lying in the same subspace. Self-expressive methods have trouble correctly clustering when the percentage of missing data is high, the matrix is high-rank, or when subspaces are close to each other.

There exists another family of methods where subspace estimation and assignment is done alternatively. Yang et al. [122] proposed to apply a matrix completion algorithm to recover the missing entries in X and then solve (3.1). This approach is likely to fail as soon as the data matrix is high-rank, i.e., $\sum_{i=1}^K r_i \approx d$. Lane et al. [68] proposed to repeatedly alternate between subspace clustering and group wise low-rank matrix completion (gLRMC). Balzano et al. [8] use GROUSE [7] 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, subspace estimation process is often faulty when an estimated cluster has points from multiple subspaces. In an extensive empirical evaluation of existing SCMD algorithms, Lane et al. [68] concluded that zero-filled elastic

net subspace clustering method [123] when alternated with low-rank matrix completion showed the overall best performance. This method is referred to as Alt-PZF-EnSC+gLRMC. A disadvantage of Alt-PZF-EnSC+gLRMC is that it requires setting two regularization hyperparameters.

Some methods instead of alternating between subspace estimation and assignment, pose these two problems in a joint optimization framework, often resulting in complex, nonconvex problems [75, 38, 44]. Matrix factorization approaches have also been adopted to SCMD [93, 95]. Both of these methods require an estimate of the subspace dimension which might not be known beforehand in several cases. Empirical experiments in [68] showed that these methods are outperformed by the alternating methods in terms of clustering error.

MILP based methods Mixed integer linear programming (MILP) based methods have not been studied for SCMD. They have been studied for fully observed data but even for that case, the proposed methods do not fully exploit the capabilities of mixed integer linear programming tools. Lazic et al. [70] were the first to propose an integer programming based method for subspace clustering called Facility Location for Subspace Segmentation (FLoSS). FLoSS generates the candidate subspaces by random sampling and then formulates the subspace clustering problem as an integer program where the goal is to minimize the orthogonal distances of data points to candidate subspaces such that it selects K subspaces, and assigns each vector to a selected subspace. Lee and Cheong [72] extended the FLoSS model to Minimal Basis FLoSS, referred to as MB-FLoSS whose subspace hypothesis generation strategy is based on finding the minimal basis subspace representation for the data matrix and relies on Low Rank Representation (LRR)[79].

Hu et al. [56] proposed the concept of constrained subspace model. They integrated facility-based model with manifold and spatial regularity constraints to develop a constrained subspace modeling framework. It is worth noting that in their experiments, the number of candidate subspaces is small (≤ 50). 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 [79]. In particular, with the help of over segmentation, they deliberate generate more number of subspaces than ground truth (e.g, $2 \times K$) with LRR, and then use integer programming to select K of them.

None of the above facility-location based approaches account for the 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 on other subspace clustering algorithms for generating candidate subspaces. Hence, these methods are incapable of correcting themselves based on the clustering quality.

Contributions

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 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 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. MISS-DSG can then identify new candidate subspaces dynamically through the use of column generation. 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 without requiring any hyperparameter tuning when number of subspaces and subspaces dimensions are known. Our computational results reveal that the proposed method can achieve higher clustering accuracy than state-of-the-art methods when data is of high-rank, the percentage of missing data is high, or subspaces are close to each other. Casting SCMD as an integer program 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.

3.2 Integer programming formulations

We observe a real-valued data matrix $X_\Omega \in \mathbb{R}^{d\times n}$ whose columns are concentrated near a union of K subspaces with dimensions $r_1, r_2, \ldots, r_K$. In Section 3.2.1, we assume that subspaces dimension $r_1, \ldots, r_K$ are known. We relax this assumption in Section 3.2.2 and let the model self-determine the subspaces dimensions with the help of a regularized objective. Data matrix X is low-rank when $\sum_{k=1}^{K} r_k \ll \min\{d, n\}$ and full-rank when $\sum_{k=1}^{K} r_k \approx \min\{d, n\}$. For an integer T, we denote $[T] := \{1, 2, \ldots, T\}$, data vector j as X_j, and data corresponding to a subset of vectors in $S \subseteq [n]$ with $X(S)$, i.e., $X(S) = \{X_j : j \in S\}$. 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.

Our approach is based on iteratively building a (potentially very-large) collection of candidate subspaces. Integer programming 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. Here r_t denotes the dimension of subspace t. We define the distance of vector $X_j, j \in [n]$ to a candidate subspace $t \in [T]$ with d_{jt}, its squared residual distance on the observed entries:

$$d_{jt} := \min_{v \in \mathbb{R}^r} \Big\{ \sum_{i:(i,j)\in\Omega} (X_{ij} - (U_t v)_i)^2 \Big\}. \tag{3.4}$$

An advantage of (3.4) is that it has a closed-form solution in terms of a simple projection operator [8]. 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 d_{jt} can be obtained as

$$d_{jt} = \|X_{\Omega,j} - P_{(U_t)_{\Omega,j}}(X_{\Omega,j})\|_2^2. \tag{3.5}$$

For fully-observed data, this is a natural choice for cost function since its value is zero if vector j lies exactly on candidate subspace t. However, with missing data, the choice of cost function becomes less clear since zero residual on observed entries for d_{jt} does not necessarily imply that vector j lies perfectly on subspace t. Balzano et al. [8] showed that for a given fully observed vector $X_j \in R^d$, if

$$\|X_j - P_{U_0}(X_j)\| < \|X_j - P_{U_t}(X_j)\| \; \forall t \in [T]\backslash\{0\}, \tag{3.6}$$

then with high probability $(1 - 4(K-1)\delta)$, for the same data vector X_j but now partially observed, if $|X_{\Omega,j}| \geq \frac{8}{3} d \max_{t \neq 0} \big((\max_{i \in [d]} \|P_{U_t}(e_i)\|_2^2) \log(\frac{2r_t}{\delta}) \big)$, then

$$\|X_{\Omega,j} - P_{(U_0)_{\Omega,j}}(X_{\Omega,j})\| < \|X_{\Omega,j} - P_{(U_t)_{\Omega,j}}(X_{\Omega,j})\| \; \forall t \in [T]\backslash\{0\}. \tag{3.7}$$

Here $\delta > 0$ is a confidence parameter and e_i is the i^{th} canonical basis vector. This implies that with high probability, subspace assignment based on (3.6) is the same as the one based on (3.7). We refer reader to [8] for more details. We assume that this property holds throughout in this paper. We also point out that a different cost model could also be incorporated into our framework.

Next, we first describe a model based on selecting K subspaces from a given collection

of $[T]$ subspaces for both known and unknown subspaces dimensions in Section 3.2.1 and 3.2.2 respectively. Next, in Section 3.3.1, 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 3.3.2, we discuss how to generate new candidate subspaces dynamically with a column generation approach. We finally discuss our unified framework MISS-DSG in Section 3.3.3.

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 that indicates whether 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 one of the open facilities. In our SCMD formulation, subspaces play the role of facilities, and vectors play the role of customers. We point out that the integer-programming methods proposed in literature also have the similar facility-location based structure [70, 72, 56]. However, our model has some key differences:

1. We account for missing data while existing integer-programming approaches restrict to fully observed data.

2. Our framework generates subspaces dynamically while existing approaches are heavily dependent on initialization.

3. Our framework is capable of handling a larger number of candidate subspaces than existing methods through the use of Benders decomposition.

For improved readability, we first discuss the formulation for the simpler case in Section 3.2.1 where subspaces dimensions are assumed to be equal, i.e., $r_1 = r_2, \ldots, r_k = r$, and r is known beforehand. We then relax this assumption in Section 3.2.2 where subspaces dimensions can be different, and are not known in advance. Moreover, this formulation does not require the number of subspaces to be known either.

3.2.1 Known subspaces dimension

Given T candidate subspaces each of dimension r, we formulate the SCMD problem as an integer program. Our complete integer programming formulation for the known subspaces

dimension case is the following [70, 56]:

$$\min_{x\in\{0,1\}^{n\times T}, z\in\{0,1\}^T} \sum_{t\in[T]}\sum_{j\in[n]} d_{jt}x_{jt} \tag{3.8a, MILP}$$

$$\sum_{t\in[T]} x_{jt} = 1, \quad \forall j \in [n] \tag{3.8b}$$

$$x_{jt} \leq z_t, \qquad \forall j \in [n], t \in [T] \tag{3.8c}$$

$$\sum_{t\in[T]} z_t = K. \tag{3.8d}$$

The objective (3.8a) ensures that the model looks for the least cost assignment of vectors and subspaces. Constraint set (3.8b) ensures that each vector is assigned to exactly one subspace, and constraints (3.8c) enforce that a vector is assigned to only a selected subspace. Constraint (3.8d) ensures that exactly K subspaces are selected. The major caveat in (3.8) compared to the models proposed in [70, 56] is that the distance metric d_{jt} in (3.8) is based on partial assignment cost (3.5).

3.2.2 Unknown subspaces dimension

A common occurrence in SCMD problems is that the number of subspaces and their dimension are unknown. In such cases, an inherent problem with model (3.8) is that there can be multiple subspaces that can fit a give set of data. For instance, consider an instance where the true rank of underlying subspaces is unknown. In this case, one might recover subspaces of rank higher than the ground truth since higher rank subspaces can have lower residuals, hence leading to an incorrect model selection. In such cases, we also need to account for complexity measure of a candidate subspace t in (3.8). We propose making use of the effective dimension (ED) to get the best union of subspaces fit in such cases. Huang et al. [57] defined effective dimension for X and a union of subspace models $S = \cup_{k=1}^{K} S_k$ as follows.

$$ED(X, S) = \frac{1}{n}\sum_{k=1}^{K} r_k(d - r_k) + \frac{1}{n}\sum_{k=1}^{K} n_k r_k \tag{3.9}$$

First term in definition of ED in (3.9), $r_k(d - r_k)$ is the complexity of U_t given by the total number of real numbers needed to specify a k dimensional subspace S_k in $\mathbb{R}^d$. Second term of (3.9), $n_k r_k$ is the total number of real numbers needed to specify the r_k coordinates of the n_k sample points in the subspace S_k.

We penalize the ED term in our objective as follows:

$$\min_{x\in\{0,1\}^{n\times T}, z\in\{0,1\}^T} \sum_{t\in[T]}\sum_{j\in[n]} d_{jt}x_{jt} + \frac{\lambda}{n}\sum_{t\in T}\Big(r_t(d-r_t)z_t + \sum_{j\in[n]} r_t x_{jt}\Big) \tag{3.10a}$$

$$\sum_{t\in[T]} x_{jt} = 1, \qquad \forall j \in [n] \tag{3.10b}$$

$$x_{jt} \le z_t, \qquad \forall j \in [n], t \in [T] \tag{3.10c}$$

$$\sum_{t\in[T]} z_t = K. \tag{3.10d}$$

Formulation (3.10) can handle subspaces of multiple dimensions and choose the best union of subspaces model by self-determining dimensions of subspaces. Constraint (3.10d) is included but can be removed if K is unknown. An important consideration in model (3.10) is the choice of regularization parameter λ which accounts for the trade-off between lower assignment cost and complexity of selected subspaces. A smaller value of λ would promote model (3.10) to select higher complexity subspaces (basis with higher dimensions r_t) while a larger value of λ would promote model to select lower complexity subspaces (basis with lower dimensions r_t). We discuss this in more detail in Section 3.5.5. Note that (3.8) is a special case of (3.10) with $\lambda = 0$.

3.3 Decomposition algorithm

In this section, we discuss how to efficiently solve formulation (3.10) without explicitly enumerating every possible candidate subspace. We generate new candidate subspaces dynamically based using column generation. We also discuss a Benders decomposition approach to handle a large number of candidate subspaces in (3.10).

The formulation (3.10) is solved via the well-known branch-and-bound method which relies on solving a sequence of (branch-constrained) LP relaxations. The LP relaxation of (3.10) 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 (3.10). 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. 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 [45]. In Section 3.3.1, we discuss a problem-specific implementation of the Benders decomposition method for the solution of the LP relaxation to (3.10). Section 3.3.2 describes how to dynamically generate improved candidate subspaces.

3.3.1 Row generation

Benders decomposition is a well-known technique to solve large LP problems that have special structure [14]. It has been applied to large-scale facility locations by Fischetti et al. [45], and our SCMD formulation (3.10) has the same structure. We apply a similar reformulation to model proposed in (3.10). 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 (3.10) is

$$\min_{w \in R^n, z \in [0,1]^T} \sum_{j \in [n]} w_j + \frac{\lambda}{n} \sum_{t \in T} r_t (d - r_t) z_t \tag{3.11a}$$

$$w_j \geq \Phi_j(z), \qquad \forall j \in [n] \tag{3.11b}$$

$$\sum_{t \in [T]} z_t = K. \tag{3.11c}$$

The function $\Phi_j(\cdot)$ 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, for each $z \in [0,1]^T$ and $j \in [n]$, $\Phi_j(z)$ is calculated by the following Benders *subproblem*:

$$\Phi_j(\hat{z}) = \min_x \Big\{ \sum_{t \in [T]} (d_{jt} + \frac{\lambda}{n} r_t) x_t : \sum_{t \in [T]} x_t = 1, 0 \leq x_t \leq \hat{z}_t, \forall t \in [T] \Big\}. \tag{3.12}$$

The function $\Phi_j(\cdot)$ is piecewise-linear and convex, and Benders decomposition works by dynamically building up a lower-bounding approximation to $\Phi_j(\cdot)$. For the ease of notation, we refer to $c_{jt} = d_{jt} + \frac{\lambda}{n} r_t$. The optimization problem (3.12) used to evaluate $\Phi_j(\hat{z})$ 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 $t_j^* := \min\{t : \sum_{s=1}^{t} \hat{z}_{\sigma_s^j} \geq 1\}$ be the *critical index*. In other words, *critical index* is index of the costliest subspace to which any portion of vector j is assigned. As described in [45], the Benders cut that can be used to lower-approximate

the function $\Phi_j(\cdot)$ is

$$w_j + \sum_{i=1}^{t_j^*-1} (c_{j\sigma^j_{t_j^*}} - c_{j\sigma^j_i}) z_{\sigma^j_i} \geq c_{j\sigma^j_{t_j^*}}. \tag{3.13}$$

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 $c^*_{ji} = c_{j\sigma^j_{t^*_{ji}}}$ denote the critical cost for the j^{th} vector in cut $i \in [p_j]$. The Benders *master problem* is then

$$\min_{w,z} \sum_{j\in[n]} w_j + \frac{\lambda}{n} \sum_{t\in[T]} r_t(d - r_t) z_t \tag{3.14a}$$

$$w_j + \sum_{\ell=1}^{t^*_{ji}-1} (c^*_{ji} - c_{j\sigma^i_\ell}) z_{\sigma^j_\ell} \geq c^*_{ji}, \forall j \in [n], i \in [p_j], \tag{3.14b, α_{ji}}$$

$$\sum_{t\in[T]} z_t = K, \tag{3.14c, β}$$

$$0 \leq z_t \leq 1, \qquad \forall t \in [T]. \tag{3.14d, μ_t}$$

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 3.3.2. For $T > K$, LP (3.14) is feasible and bounded, and hence an optimal solution $(\hat{w}, \hat{z})$ exists. The subproblem (3.12) is solved to evaluate $\Phi_j(\hat{z})$ for each $j \in [n]$, and to generate new Benders cuts (3.13). 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 (3.14). The Benders procedure stops when no new cuts are added. At this point, the LP relaxation of (3.8) is solved.

3.3.2 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 (3.10). Column generation is a classical method for solving large-scale LP [48] that also has seen significant use in solving MILP problems [12].

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 (3.14), 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 may decrease. 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 set $[T]$ of candidate subspaces is optimal and no new candidate subspaces need to be included in $[T]$.

Given the optimal dual variables $(\hat{\alpha}, \hat{\beta})$ to the solution of (3.14), the reduced cost of a column/subspace variable z_t is given by the formula

$$\frac{\lambda}{n} r_t(d - r_t) - \sum_{j \in [n]} \sum_{i \in [p_j]} \hat{\alpha}_{ji} \max\{c^*_{ji} - c_{jt}, 0\} - \hat{\beta}. \tag{3.15}$$

The max operator in (3.15) appears from the fact that for subspace t and data vector $j \in [n]$, $(c^*_{ji} - c_{jt})z_t$ is considered in summation in constraint (3.14b), only if c_{jt} is smaller than the critical cost c^*_{ji}.

Recall (3.4), that describes the assignment cost as a function of the basis matrix

$$c_{jt} := h_j(U_t) := \min_{v \in \mathbb{R}^r} \Big\{ \sum_{i:(i,j) \in \Omega} (X_{ij} - (U_t v)_i)^2 \Big\} + \frac{\lambda}{n} r. \tag{3.16}$$

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

$$\min_U g(U) = \min \frac{\lambda}{n} * r(d - r) - \sum_{j \in [n]} \sum_{i \in [p_j]} \alpha_{ji} \max\{c^*_{ji} - h_j(U), 0\} \tag{3.17}$$

Here r denotes dimension of the basis U. We next point out that problem (3.17) is not a convex optimization problem, and hence even for a fixed subspace dimension r, it is difficult to solve to provable global optimality. We find locally minimal solutions to (3.17) with a gradient-based approach. To handle different dimensions, in each iteration of column generation, we solve (3.17) for $r_U \in \{1, 2, \ldots, r_{max}\}$ where r_{max} is an upper bound on the subspace dimension and is provided as an input parameter to the model.

Gradient-based approach for pricing problem. We solve pricing problem (3.17) for $r \in \{1, 2, \ldots, r_{max}\}$ with $U \in R^{d \times r}$. Observe that If $h_j(U) \neq c^*_{ji}\ \forall j \in [n], i \in [p_j]$, then the function $g(U)$ is differentiable. The partial derivative of $g(\cdot)$ with respect to matrix element U_{ab} evaluated at current iterate $\hat{U}$ is given by

$$\frac{\partial g(\hat{U})}{\partial U_{ab}} = - \sum_{j \in [n]} \sum_{\substack{i \in [p_j]: \\ c^*_{ji} - h_j(\hat{U}) > 0}} 2\alpha_{ji} \sum_{\ell \in \Omega_j} (X_{\ell j} - \hat{u}_\ell^\top \hat{v}_j) \hat{v}_{jb} \quad \forall a \in [d], b \in [r]. \tag{3.18}$$

With condition $c^*_{ji} - h_j(\hat{U}) > 0$, we implicitly use a subgradient of 0 at points of non-differentiability. Here $\hat{u}_\ell$ represents ℓ^{th} row of basis $\hat{U}$ and $\hat{v}_j$ is the minimizer in (3.16) for $U_t = \hat{U}$. We denote $\nabla g(U) = \begin{bmatrix} \frac{\partial g(U)}{\partial U_{11}} & \cdots & \frac{\partial g(U)}{\partial U_{1r}} \\ \vdots & \vdots & \vdots \\ \frac{\partial g(U)}{\partial U_{d1}} & \cdots & \frac{\partial g(U)}{\partial U_{dr}} \end{bmatrix}$.

Algorithm 3: 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 $m = 0, \hat{U} = U_0, \mathcal{U} = \{\}$; `/* iteration count */`

2 **while** *m<maxIt and* $\|\nabla g(\hat{U})\| > \epsilon$

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

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

5 $c_{jm} = \|(X_j)_\Omega - \hat{U}_{\Omega,j}\hat{v}_j\|_2^2$;

6 **end**

7 Calculate $\nabla g(\hat{U})$ using (3.18) ; `/* Requires` $\hat{U}, \hat{v}_j \ \forall j \in [n]$ `*/`

8 Calculate $\tilde{g}$ using (3.19) ;

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

10 $\hat{U} \leftarrow \hat{U} - \gamma\nabla g(\hat{U})$; `/* move in negtaive gradient direction */`

11 $m \leftarrow m + 1$;

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

13 **end**

Output: Subspaces basis $\mathcal{U}$

We outline our gradient-based approach for locally solving pricing problem (3.17) for a fixed subspace dimension r in Algorithm 3. Given that we use a gradient-based approach to the nonconvex problem (3.17), we solve the pricing problem multiple times with different random choices of U_0 to identify different locally-optimal solutions. For fully-observed data, $r+1$ vectors per subspace are necessary and sufficient for subspace clustering. Hence, we randomly sample N($> r+1$) vectors from costliest M vectors in current LP solution of (3.14). We let $M > N$ in order to add randomness in initialization and let $M = 5r_{max}$ and $N = 2r$. This choice helps to select a subset of vectors with high residuals in current solution and initialize gradient descent algorithm with a best-fit subspace on that subset of vectors. Then, we use a fast low-rank matrix completion algorithm, e.g. GROUSE [7], 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 3 (line 3).

To calculate gradient using equation (3.18), we first need to project each vector on observed entries ($X_{\Omega,j}$) to $\hat{U}$. The corresponding projection matrix $(\hat{U}^\top_{\Omega,j}\hat{U}_{\Omega,j})^{-1}\hat{U}_{\Omega,j}$ is

unique to each vector since each vector j is observed on a different subset of dimensions, i.e. $\Omega_j \subseteq \{1, 2, \ldots, d\}$. Thus, each iteration of gradient descent requires calculating n projection operators (lines 3-6 in Algorithm 3). We then calculate gradient in line 7.

An important choice to converge to local optima in gradient based iterative methods is the choice of *step size*. In recent works, adaptive methods have become wide-spread and have proved to beneficial. We use the Polyak step size [97]. Exact Polyak step requires optimal value of objective function. Since optimal value g^* is unknown, we approximate it with $\tilde{g}$ (in line 8) as follows

$$\tilde{g} \approx \frac{\lambda}{n} r(d-r) - \sum_{j \in [n]} \sum_{\substack{i \in [p_j]: \\ c^*_{ji} - h_j(\hat{U}) > 0}} \alpha_{ji} c^*_{ji}. \tag{3.19}$$

This choice of $\tilde{g}$ is reasonable since the perfect union of subspaces model for X_Ω would imply zero residuals i.e., $c_{jt} = h_j(U_t) = 0$ in (3.17).

Polyak step size is then calculated as $\gamma \leftarrow \frac{\tilde{g} - g(\hat{U})}{\|\nabla g\|_2^2}$. To avoid step size 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) . Empirical experiments show that this choice of step size works well and converges faster than a decaying step size. We discuss this in more detail in Section 3.4.3.

With gradient and step size information at hand, we finally take a gradient step (line 10). We terminate gradient descent when gradient becomes too small ($\|\nabla(g)\| < 0.001$) or we reach the maximum number of allowed iterations (500 in our case). We store all the iterates generated during gradient descent approach (line 12) since each iterate is a potential candidate subspace to be added to master problem (3.14). Of all the columns generated, only those with negative reduced cost as calculated in (3.15) are added to the master problem (3.14).

3.3.3 MISS-DSG: Mixed Integer Subspace Selector with Dynamic Subspace Generation

We now have all the tools to develop a unified MILP framework, MISS-DSG, for SCMD problem that integrates the use of Benders decomposition and column generation. We first point out that we generate new columns (z_t variables) only at root node in branch-and-bound search tree. We describe MISS-DSG framework in Algorithm 4. We initialize the algorithm with m randomly generated subspaces for each possible low-dimension (line 1) to initialize model (3.14). We then solve the master LP relaxation (3.14) in line 9, and generate Benders cuts for each $j \in [n]$ (lines 10-14). Observe that if $\Phi_j(\hat{z}) = \hat{w}_j$, then the

Algorithm 4: Mixed Integer Subspace Selector with Dynamic Subspace Generation

Data: X_Ω
Input: max rank r_{max}, min and max # of multi-starts: $\eta_{\min}, \eta_{max}$, max # iterations $i_{\max}$
Output: Clusters K, dimensions $r_k \; \forall k \in \{1, 2, \ldots, K\}$, Segmentation of $[n]$ in K clusters: S_k

Initialize MILP model (3.10) with $[T]$ as m random subspaces of rank $r_i \; \forall i \in \{1, 2, \ldots, r_{max}\}$;
Calculate $c_{jt} \; \forall j \in [n], t \in [T]$;
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`
    while *generate cuts*
        generate cuts $\leftarrow False$;
        solve master LP relaxation (3.14) 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 (3.13) to master (3.14);
                generate cuts $\leftarrow True$
        end
    end
    `// generate new columns`
    for $r = 1, \ldots, r_{max}$ **do**
        for $\eta = 1, \ldots, \eta_{\max}$ **do**
            $U_0 \leftarrow$ BFS on randomly sampled $2r$ vectors from $5r_{max}$ vectors with largest $\hat{w}_j$;
            $\mathcal{U} \leftarrow$Solve pricing problem using Algorithm 3 to generate candidate subspaces ;
            $[T'] \leftarrow$Negative reduced cost columns in $\mathcal{U}$ **if** $[T'] \neq \emptyset$
                $[T] \leftarrow [T] \cup [T']$; `/* Add new` $z_t, t \in [T']$ `variables */`
                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 (3.14) ; `/* invalid due to new` z_t `vars */`
end
$\hat{x_{jt}}, \hat{z}_t \leftarrow$ Solve MILP model (3.11) with $z_t \in \{0, 1\}^T$, give a callback routine for Benders cuts ;
return $\{S_t = \{j \in [n] : \hat{x}_{jt} = 1\}, U_t, \forall t \in [T] \text{ s.t. } \hat{z}_t = 1\}$;

generated inequality does not improve the approximation to $\Phi_j(\cdot)$, and the cut is not added to (3.14) (line 11). We repeat this until no violated cuts are found.

We next proceed to generate new columns (lines 16-26) by solving the pricing problem (3.17) in order to look for negative reduced cost columns. We solve pricing problem for every possible dimension $r \in \{1, 2, \ldots, r_{max}\}$ (line 16). For each dimension r, we do multi-start (line 17) and locally solve the pricing problem (3.17) using Algorithm 3 and store all generated iterates in $\mathcal{U}$ (line 19). The negative reduced cost columns in $\mathcal{U}$ (lines 20-20), are added to the master LP (line 21). We put lower and upper bound on number of multi-starts, $\eta_{\min}$ and $\eta_{\max}$ respectively. We perform a minimum of $\eta_{\min}$ multi-starts and proceed to next dimension if negative reduced cost columns are found (lines 22-24). We perform a maximum of $\eta_{\max}$ multi-starts to look for negative reduced cost columns and proceed to next dimension after that. In our experiments, we use $\eta_{\min} = 5$ and $\eta_{\max} = 15$.

If new columns are found in column generation process, we delete existing Benders cuts since they become invalid due to new z_t variables (line 28), and return to the process of generating Benders cuts. We repeat cut generation (lines 7-15) and column generation (lines 16-26) as long as we are able to generate new columns or till we reach the maximum iteration limit, $i_{\max}$ which we set to be 15.

If we fail to find a negative reduced cost column or reach maximum allowed iterations, we exit the root node loop and pass the updated MILP model with the new columns and cuts included to a MILP solver (line 30). We use Gurobi 8.1 as our MILP solver. Within the MILP branch-and-cut framework, master solution $(\hat{w}, \hat{z})$ is generated by primal heuristics or when current-node solutions happens to be integral. We need to check the solution validity before letting the solver update the incumbent. To handle this, we pass a `lazy constraint callback` that certificates validity of current integral solution $(\hat{w}, \hat{z})$. In particular, whenever an integer solution is found during the branch-and-bound tree, `lazy constraint callback` is invoked and it either returns one or more valid cuts that prevents the current incumbent update or certifies the validity of current solution $(\hat{w}, \hat{z})$. Finally, we use the optimal MILP solution to determine the selected subspaces and mapping of each vector to a selected subspace (line 31). In particular, we select subspaces $t \in [T]$ such that $\hat{z}_t = 1$. Each vector is naturally assigned to it's closest subspaces among the selected subspaces.

3.4 Algorithmic choices

In this section, we justify various algorithmic choices through empirical experiments.

3.4.1 Experimental setup

For our experiments, we use Gurobi 8.1 as the MILP solver. Additionally, 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 use Python for running all the experiments reported in this paper. In this Section, we consider randomly generated instances, similar to [68]. 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 random $k \in [K]$ and $v_j \in \mathbb{R}^{r_k}$ is sampled from a standard Gaussian. After generating data matrix X, we uniformly at random drop a percentage f of the entries in X.

3.4.2 Impact of Benders decomposition

MISS-DSG requires solving LP relaxation repeatedly as new columns are added at the root node. Benders decomposition gives a significant speedup in the time required to solve the LP. To demonstrate, we compare solution times for solving LP relaxation with and without use of Benders decomposition. We generate the identical set of subspaces ($[T]$) for both the formulations and solve the resulting LP relaxations. In particular, without Benders refer to directly solving LP relaxation of (3.8) while with Benders refers to solving (3.14). We point out that for $\lambda = 0$ (3.14) is equivalent to (3.8).

We first consider the effect of number of subspaces ($|T|$) in Table 3.1. All results reported in Tables 3.1 and 3.2 are averaged over five different random trials. We fix $d = 30, n = 200, K = 6, f = 0, r_k = 3\ \forall k \in [K]$ and vary $|T|$. As $|T|$ increases from 100 to 4000, the LP relaxation time for (3.8) grows from 1.2s to 1380.3s. On the other hand, with Benders approach (3.14), the LP relaxation time grows from 0.1s to 3.1s only. Thus, Benders approach give a speed up of $5\times$ for $|T| = 100$ and $460\times$ for $|T| = 5000$.

We next fix $d = 30, K = 6, f = 0, |T| = 500, r_k = 3\ \forall k \in [K]$ and vary n as shown in Table 3.2. We observe a similar behavior as before. In particular, Benders approach gives a speed up of $15\times$ for $n = 100$ and $255\times$ for $n = 1200$. Since MISS-DSG requires solving the LP relaxation iteratively whenever a new column is identified, Benders decomposition gives a significant boost in computational performance specially when $n * |T|$ is large.

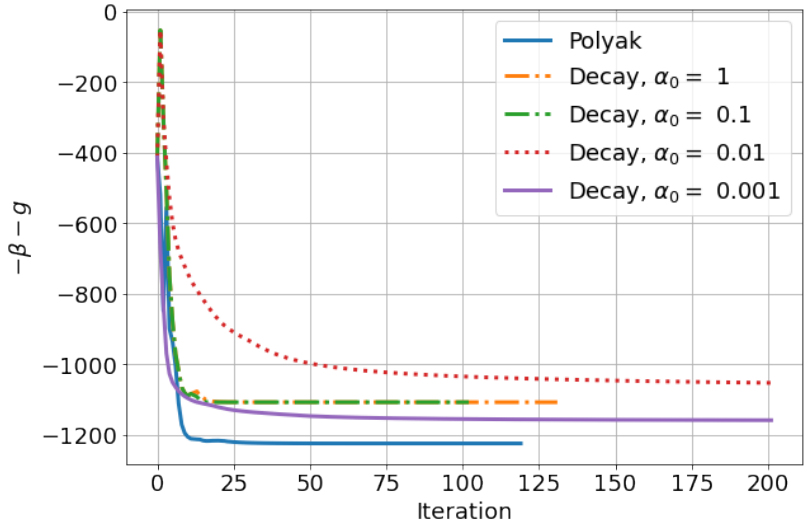
Table 3.1: Effect of number of subspaces $|T|$ on LP relaxation time (s) for $d = 30, n = 200, K = 6, f = 0, r_k = 3\ \forall k \in [K]$

$\|T\|$	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 3.2: Effect of number of vectors n on LP relaxation time (s) for $d = 30, K = 6, f = 0, |T| = 500, r_k = 3\ \forall k \in [K]$

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

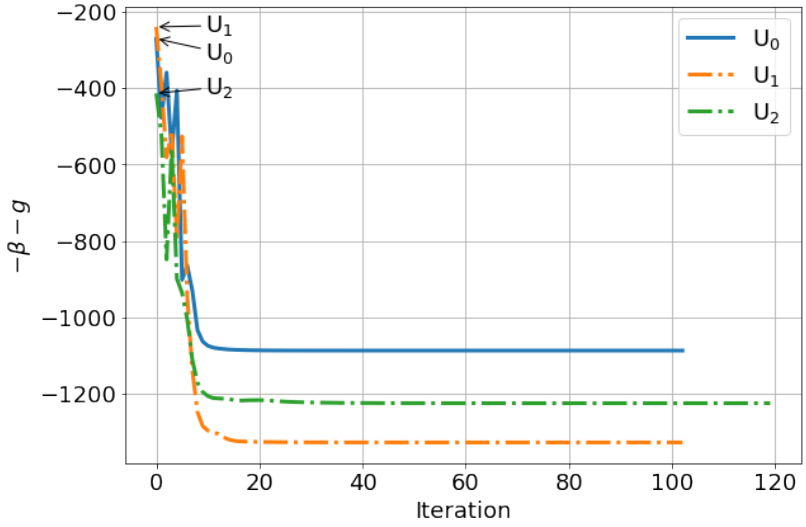
Figure 3.1: Comparison of Polyak step size with decay step size for same starting point



3.4.3 Why Polyak step size?

For the pricing problem (3.17), we can approximate the optimal value of function by letting the assignment cost for new subspace to be 0, i.e., perfect subspace recovery. With an estimate of the optimal value of function at hand, we can now use Polyak step size [97]. A major advantage of Polyak step size is that it is adaptive in nature and does not require tuning the initial step size, often required in constant or diminishing step size rules. Moreover, empirical experiments indicate that the problem converges to local solution in fewer iterations compared to the standard diminishing approaches. We consider a test instance with $d = 20, n = 200, K = 5, f = 35\%, r_k = 4\ \forall k = 1, \ldots, 4$. We let number of subspaces and their rank to be an input to the model, and hence we do not consider the regularization term in (3.15), i.e., we let $\lambda = 0$. We compare convergence rate for Algorithm 3 with Polyak and decaying step size $(\frac{\alpha_0}{\text{iter}})$ where iter is the iteration number as shown in Figure 3.1. We plot reduced cost $(-\beta - g)$ (3.15) on y-axis and iteration number on x-axis.

Figure 3.2: Algorithm 3 converges to different local solutions for different starting points



For decaying step size, we consider initial step size $\alpha_0 \in \{0.001, 0.01, 0.1, 1\}$. We observe that Polyak step size leads to the fastest convergence. Moreover, since Polyak approach does not require training the initial learning rate, we find it computationally more economical over the decaying step size approach. The performance of this one instance is indicative of the general behavior.

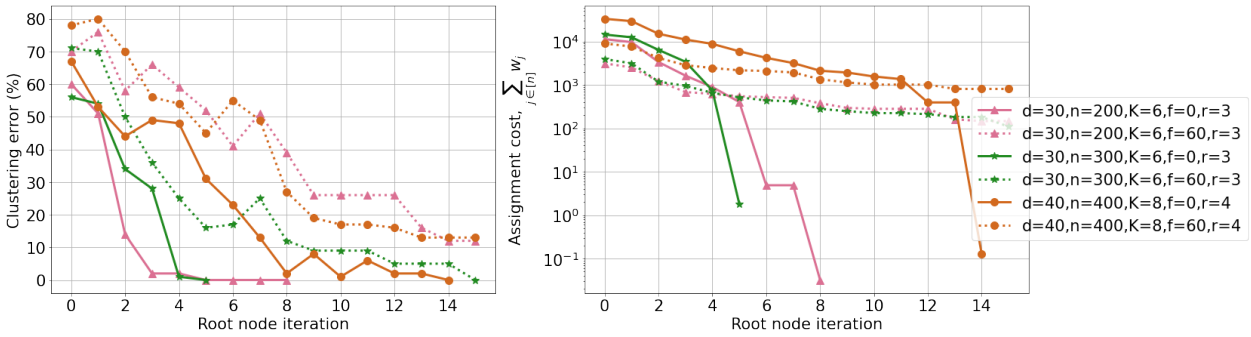
3.4.4 Why multi-start?

We next discuss the importance of doing multi-start for solving the pricing problem (3.17). We consider the same instance as discussed in Section 3.4.3 and solve the pricing problem for different initialization points. As shown in Figure 3.2, we observe that three different choices lead to three different local solutions. Hence, multi-starting can help the algorithm from getting stuck at a *bad* local minima (which might have positive reduced cost) and identify subspaces with negative reduced cost.

3.4.5 Value of solving pricing problem

A major drawback in existing approaches which try to recover the underlying subspaces basis has been over reliance on good initialization [8, 72, 56]. We demonstrate that MISS-DSG is robust to initialization and recovers underlying subspaces accurately even with random initialization. We let $r_k = r \; \forall k \in [K]$ and assume that both K and r are known, and thus we fix $\lambda = 0$ in Algorithm 4. We demonstrate evolution of clustering quality with addition of new columns generated from solving pricing problem in Figure 3.3. We consider

Figure 3.3: Evolution of clustering in MISS-DSG. Clustering error on the left and assignment cost (log-scale) on the right. Solid lines represent fully-observed data and dotted lines represent $f = 60\%$ missing data.



randomly generated instances similar to Section 3.4.3 and initialize MISS-DSG with 500 randomly sampled subspaces from $Gr(r, d)$. We consider fully observed data ($f = 0$) and $f = 60\%$ missing data case for three different instances, $(d = 30, n = 200, K = 6, r = 3)$, $(d = 30, n = 300, K = 6, r = 3)$, and $(d = 40, n = 400, K = 8, r = 4)$, as shown in Figure 3.3. We report clustering error and assignment cost $(\sum_{j\in[n]} w_j)$ against root node iteration in Figure 3.3. By root node iteration, we refer to an iteration of generating new candidate subspaces and Benders cut, i.e., the outer-most loop in Algorithm 4. To avoid stalling, we allow at most 15 root node iterations. We first point out that although assignment cost is a non-increasing function of root node iteration clustering error is not. The goal of Algorithm 4 is to generate subspaces with lower assignment cost and hence addition of new columns can only improve the objective value of (3.14). It is possible that addition of new columns reduces the assignment cost but marginally increases the clustering error. However, as Algorithm 4 progresses and pricing problem is solved subsequently, the generated subspaces stabilize and get closer to the ground truth. Hence clustering error has a downward trend as shown in Figure 3.3. We also highlight that for fully-observed instances, clustering error reduces from $\approx 60\%$ to $\approx 0\%$ and assignment cost reduces from $\approx 10^4$ to $\approx 10^0$. Similarly, for instances with 60% missing data, clustering error reduces from $\approx 75\%$ to $\approx 10\%$ and assignment cost reduces from $\approx 10^4$ to $\approx 10^2$. Thus, adding new columns by solving pricing problem improves the clustering quality significantly. We next observe that all three instances converge faster for fully observed data than for missing data. This is expected since fully-observed instances carry more information about the underlying subspace model and hence even local solutions to the pricing problem converge to the subspaces close to ground truth faster. The assignment cost which is $< 10^0$ for fully observed data while $\approx 10^2$ for 60% missing data indicates that Algorithm 4 recovered ground truth subspaces almost perfectly in fully-observed data case but not in high-missing

data case. We point out that there are two possible reasons for getting non-zero assignment costs for noiseless data in our model: a) we solve the pricing problem (3.17) locally, and b) we generate new-columns only at the root node. However, for practical purposes, irrespective of these two challenges, Algorithm 4 recovers high quality clustering which is competitive with state-of-the-art methods.

3.5 Computational Results

In this section, we evaluate the performance of MISS-DSG against various SCMD methods from the literature. In our experiments, we use Python as the programming language and Gurobi 8.1 as the MILP solver. Additionally, we set a time limit of 5000s for each MISS-DSG run. These experiments were performed on a cluster of 4 core 16 GB machines with Xeon X5690 CPU running at 3.46GHz.

3.5.1 Synthetic dataset

The difficulty of SCMD depends on several factors such as the arrangement of subspaces, the separation between subspaces, the total rank of the data, and the percentage of the missing data. For an extensive comparison, in addition to the random instances discussed in Section 3.4.1, we also consider two types of semi-random subspace arrangements : *independent* and *disjoint*. Semi-random instances allow us to control the separation between the subspaces which is measured by the affinity. Separation plays a crucial role in recovery of the subspaces for *independent* and *disjoint* case.

Two independent subspaces: We consider two independent subspaces[1] 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 gets more challenging [108]. Similar to [1], we generate the two independent 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 identity matrix of size $r \times r$, and 0_r denotes zero matrix of size $r \times r$. Of the total n data points, we create $\frac{n}{2}$ data points from each of the two subspaces. We first randomly create data points within each of the $2r$-dimensional subspaces and then transfer them to the d-dimensional space. Let $\hat{X}_1$ and $\hat{X}_2$ denote data points generated

[1] A collection of subspaces $S_1, \ldots, S_c$ is said to be independent if $\sum_{i=1}^{c} \dim(S_i) = \dim(\cup_{i=1}^{c} S_i)$.

from each subspace within $2r$-dimensional space. $\hat{X}_1$ is created from U_1 as $\hat{X}_1 = U_1 * W$ where $W \in R^{r \times \frac{n}{2}}$ and each entry of W is sampled from Gaussian distribution, $\mathcal{N}(0, 1)$. $\hat{X}_2$ is created similarly. Data points from dimension $2r$ are then transferred to ambient dimension d by multiplying with a randomly generated orthonormal basis $P \in R^{d \times 2r}$ as $X_i = P \times \hat{X}_i \; \forall i = 1, 2$. This orthonormal projection preserves the affinity between two subspaces from $2r$-dimensional space to d-dimensional space.

Three disjoint subspaces: For disjoint synthetic subspaces[2], we generate $\frac{n}{3}$ vectors from each of the subspaces. The three initial subspaces are constructed as [1]:

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

Rest of the construction is identical to the independent case.

Random subspaces: Clustering on *independent* and *disjoint* subspaces (for high affinity) is less difficult than clustering on intersecting subspaces[3]. This is the most general subspace arrangement with no assumption on the subspaces, and any two subspaces can have a nontrivial intersection. One should observe that data points belonging to the intersection of two subspaces lead to non-unique membership assignments. Two distinct two-dimensional planes in three dimensions is an example of intersecting subspaces [1]. These random instances are generated in the similar manner as describe in Section 3.4. After generating data matrix X with either of the methods above, we uniformly at random drop a percentage f of the entries of the matrix X yielding the set of observed entries Ω.

3.5.2 Metrics

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

- Clustering Error: Let $\{G_1, G_2, \ldots, G_K\}$ be the ground truth clusters where $G_k \subseteq [n] \; \forall k \in [K]$ and similarly $\{P_1, P_2, \ldots, P_{K'}\}$ be the predicted clusters. We get the best matching between predicted and true clusters by solving an assignment problem. Note that we also allow for the case when number of predicted and true clusters are different, i.e., $K \neq K'$. We let $\mathcal{M}$ be the set of all possible assignments i.e.,

[2]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 $S_i \cap S_j = \{0\} \; \forall i \neq j$). Note that independent subspaces are disjoint but the converse is not always true.

[3]A collection of subspaces $S_1, S_2, \ldots, S_c$ is intersecting if $1 \leq \dim(S_i \cap S_j) < \min\{\dim(S_i), \dim(S_j)\}$ for some $i \neq j$.

$\mathcal{M} = \{(P_{k'}, G_k)\ \forall k \in [1, \ldots, K], k' \in [1, \ldots, K']\}$. We denote cost of matching $M = (P_{k'}, G_k) \in \mathcal{M}$ with c_M which indicates the number of disagreements between the two clusters in M, i.e., $c_M = |P_{k'} \triangle G_k|$. Solving the following model then gives the optimal matching:

$$\min_{y \in \{0,1\}^{|\mathcal{M}|}} \sum_{M \in \mathcal{M}} c_M y_M \tag{3.20a}$$

$$\sum_{M \in \mathcal{M}: G_k \in M} y_M \leq 1 \qquad \forall k \in [K] \tag{3.20b}$$

$$\sum_{M \in \mathcal{M}: P_{k'} \in M} y_M \leq 1 \qquad \forall k' \in [K'] \tag{3.20c}$$

$$\sum_{M \in \mathcal{M}} y_M = \min\{K, K'\} \tag{3.20d}$$

Objective (3.20a) minimizes the disagreements in the matching, constraints (3.20b) and (3.20c) ensure that each cluster in $\{G_1, G_2, \ldots, G_K\}$ is mapped to at most one cluster in $\{P_1, P_2, \ldots, P_K\}$, and vice-versa. Constraint (3.20d) forces the model to do the maximum number of assignments possible. Once we get the optimal matching $(\hat{y})$, we can calculate the clustering error as follows:

$$\text{Clustering error} = 100 \times \frac{\sum_{M \in \mathcal{M}} c_M \hat{y}_M}{2n}$$

Note that we divide by 2 since every vector when mismatched is penalized twice. In particular, if vector $j \in [n]$ is mismatched, then it penalized in matching (P_k, G_k), when $j \in P_k, j \notin G_k$ and also in matching (P_i, G_i) such that $j \notin P_i, j \in G_i$ and $i \neq k$.

- 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 true and recovered unobserved entries as follows:

$$\text{Completion error} = \|I_{\Omega^c} \circ (\hat{X} - X_{GT})\|_F / \|I_{\Omega^c} \circ (X_{GT})\|_F$$

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 using GROUSE [7] if the rank is known and SVT [24] if the rank is unknown.

3.5.3 Comparison against other MIP approaches

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

- FLoSS [70]: FLoSS was the first work to formulate subspace clustering problem as facility location problem. The candidate subspaces in FLoSS are initialized from the data by randomly selecting r-tuples of linearly independent points, with $2 \leq r < d$. Data corresponding to r-tuple 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. Assignment cost of vector to subspace is similar to (3.5). We do the extension to partially observed data by using the same cost model as ours, i.e., residual on observed entries (3.5). To handle missing data in subspace generation process, we perform LRMC using GROUSE on each tuple 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 r vectors. We then solve model (3.8) with $[T]$ consisting of all the subspaces generated with the above strategy. We refer this algorithm as MIP-RANDOM.

  We point out that authors in [70] 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 solving pricing problem in generating new subspaces over generating subspaces randomly. 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 [72]: MB-FLoSS is similar to FloSS with a major difference in the candidate subspaces generation strategy. Instead of doing random sampling, candidate subspace are generated by solving the following optimization problem:

$$C^* = \arg\min_{C} \|C\|_{2,1} + \gamma\|E\|_{1,2} \quad \text{s.t. } X = XC + E. \tag{3.21}$$

  Here C is the coefficient matrix with $\|C\|_{2,1} = \sum_{i=1}^{d} \sqrt{\sum_{j=1}^{n}(C_{ij})^2}$ and E is the error matrix 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 (3.21) to get C^*. Lee and Cheong [72] solve this constrained convex program (3.21) using Alternating Direction Multiplier Method (ADMM) as in [79]. We point out that optimization problem (3.21) is

similar to (3.1). However, the motivation in (3.21) is to seek the minimal basis representation, whereas in (3.1), the joint sparsity regularization was introduced to ensure connectivity in the similarity graph generated by encouraging data points from the same subspace to use common representative points. Lee and Cheong [72] instead of ensuring connectivity, perform over-segmentation on C^*. In particular, 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, and therefore only top r largest absolute value coefficients in each column are retained to form a candidate subspace. For fully-observed data, one can do SVD to get a candidate subspace. For missing data case, we perform LRMC using GROUSE to get the best fit subspace which is then used a a candidate subspace. The number of candidate subspaces generated is therefore the number of unique subspace proposed by all the data points.

- BB-LRR [56]: BB-LRR also proposed a facility-location type MIP approach but with a different assignment cost function and subspace generation process. BB-LRR generates candidate subspaces by over-segmentation in LRR [79]. They set the number of clusters larger than the ground truth in spectral clustering step, eg $K + 3$ instead of K. For assignment cost function, instead of using the the residual distance directly (d_{jt} as in (3.5)), they use the following normalized distance (d'_{jt}) by introducing rank-dependent noise level i.e,

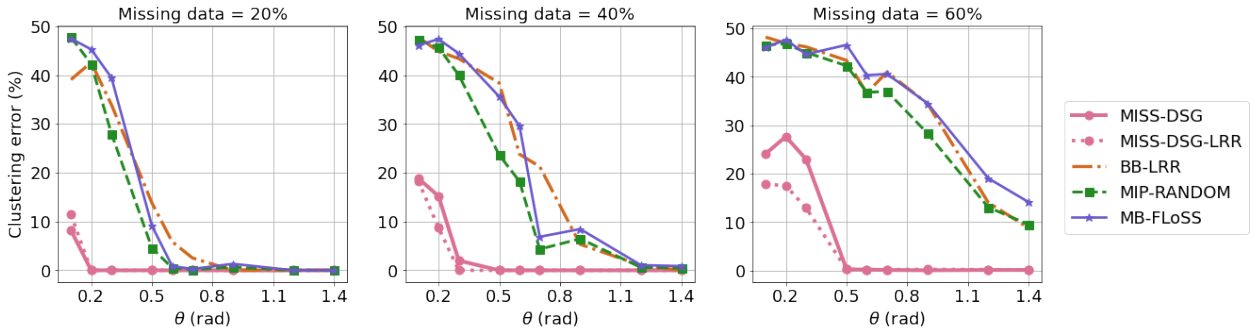
$$d'_{jt} = \frac{d_{jt}}{2\sigma^2_{(r_j)}} + \ln \sigma_{(r_j)}$$

  where $\sigma_{(r_j)}$ is estimated in the following way: first the standard deviation of each candidate subspace is estimated according to the supported points, and then for each subspace rank, the median of the K minimal deviations is selected as the estimated noise level. We point out that 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 [56]. 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.

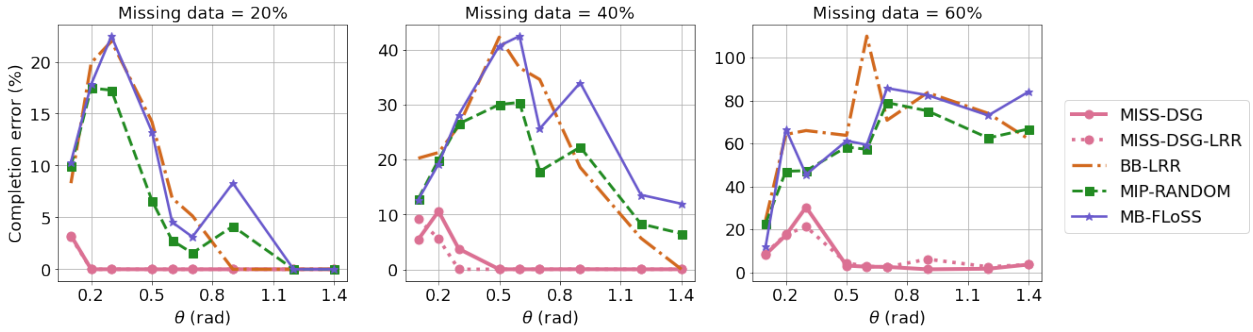
We point out that existing MIP-based methods either use randomly sampled subspaces or rely on generating subspaces with self-expressive methods, and are incapable of correcting

Figure 3.4: Performance comparison for different MIP-based methods as a function of subspace angles for two independent subspaces . Parameters are $d = 20, n = 200, K = 2$, and $r_1 = r_2 = 2$

(a) Clustering error



(b) Completion error



themselves based on the current clustering quality. We next compare the performance of MIP-based methods on independent and randomly sampled instances.

Independent 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 missing data percentage $f \in \{20, 40, 60\}$, and report clustering and completion errors in Figure 3.4. For our method, we consider both a random initialization (referred to as MISS-DSG) and an initialization with LRR similar to [56] (referred to as MISS-DSG-LRR). Existing facility location methods give similar performance for all three missing data cases as shown in Figure 3.4a. Higher value of θ implies that subspaces have low-affinity and hence clustering task is easier. 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-

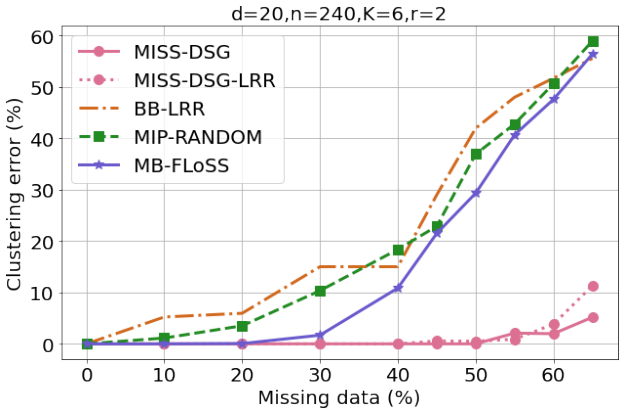
DSG still gives low clustering errors in the same regime. We observe a similar trend in completion error as shown in Figure 3.4b. For $f = 60\%$ missing data case, we observe that completion errors increases with increase in θ while one expects it to decrease. Candidate subspace generation strategy in existing MIP-based methods use self-expressiveness and low rank matrix completion, both of which fail in presence of high missing data. LRMC step for calculating completion error can often be faulty when 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 new columns between 800-4000 in these instances while MIP-RANDOM had 5000 candidate subspaces. Clearly, MISS-DSG outperforms MIP-RANDOM, thus showing the value of solving pricing problem (3.17) in generating 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 solving pricing problem (3.17) generates better candidate subspaces than LRR method.

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 missing data between 0 to 65% as shown in Figure 3.5a. The reported results are averaged over 10 random trials. We observe that MIP-RANDOM gives the lowest clustering error among existing MIP methods but it is outperformed by MISS-DSG in high missing data regime. BB-LRR and MB-FLOSS methods give low clustering errors ($< 10\%$) for $f < 30\%$ and deteriorate significantly as missing data is further increased. MISS-DSG with random or LRR initialization gives perfect clustering for $f < 60\%$. We report completion errors for these instances in Table 3.3. MISS-DSG recovers the missing data with no reconstruction error for f up to 50% while other MIP-based methods give high reconstruction errors for $f > 20\%$.

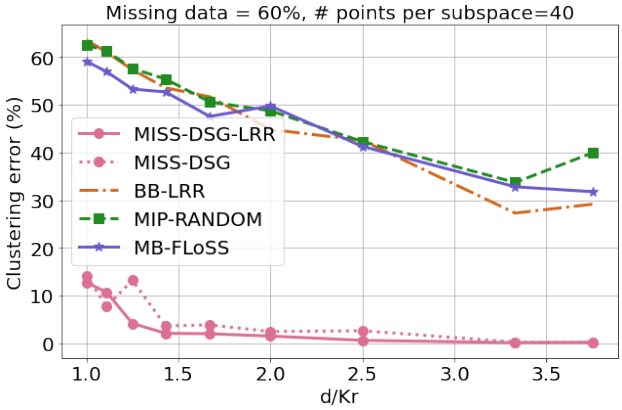
We next study the effect of total rank of the data matrix on clustering error. We generate a variety of instances with $n/K \approx 40, f = 60\%$, and vary d, K, r to get $d/Kr \in [1, 4]$. Lower ratio implies that matrix is high-rank, thus making the clustering task more difficult. Due to a high amount of missing data, we observe high clustering errors in all the existing MIP-based methods. As matrix rank gets smaller relative to the ambient dimension, clustering task becomes easier and hence the clustering error improves as shown in Figure 3.5b. MISS-DSG recovers perfect clustering when ratio is > 2 and between $0 - 10\%$ for ratio $\in [1, 2]$ as shown in Figure 3.5b. We observe that in 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

Figure 3.5: 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



completion errors, as shown in Table 3.4 for MISS-DSG for ratio $\in [1, 1.7]$.

3.5.4 Comparison against state-of-the-art methods

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 [122]. In particular, Yang et al. [122] proposed solving (3.1) with $\|\cdot\|_1$ regularization as follows:

$$C^* = \arg\min \lambda \|I_\Omega \circ (X_{ZF} - X_{ZF}C)\|_F^2 + \|C\|_1 \qquad \text{s.t. } \text{diag}(C) = 0 \tag{3.22}$$

  Coefficient matrix C^* is then processed by the spectral clustering algorithm in order to obtain data segmentation as discussed in Section 3.1. This method was found superior to the other methods proposed in [122] for SCMD.

Table 3.3: Average completion error (%) for random instances in Figure 3.5a

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 3.4: Average completion error (%) for random instances in Figure 3.5b

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

- Alt-PZF-EnSC+gLRMC: In a review article on SCMD by Lane et al. [68], alternating between elastic-net subspace clustering [123] and group low-rank matrix completion [75] 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 coefficient matrix C^* which is then processed by the spectral clustering algorithm to get data segmentation:

$$\begin{aligned} C^* = \arg\min \lambda \|I_\Omega \circ (X_{ZF} - X_{ZF}C)\|_F^2 + \zeta\|C\|_1 + (1-\zeta)\|C\|_F^2 \\ \text{diag}(C) = 0,\ C \in \mathbb{R}^{n\times n} \end{aligned} \tag{3.23}$$

  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[24], to fill the missing entries and get $\hat{X}$. In next iterations, X_{ZF} in (3.23) is replaced by $\hat{X}$, and algorithm alternates between clustering and completion for the given number of iterations.

- k-GROUSE: Balzano et al. [8] proposed an extension of well known K-Subspaces algorithm [19, 114] 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 (3.5). Given a cluster of vectors,

Table 3.5: Parameter choices for state-of-the-art methods for SCMD

Method	Parameter
EWZF-SSC [122]	$\lambda = \frac{\alpha}{\max_{i \neq j} \|(X_{ZF})_{\Omega_j}^T (X_{ZF})_{\Omega_j})\|_{ij}}$ $\alpha \in \{5, 20, 50, 100, 200, 320\}$
k-GROUSE [8]	-
Alt-PZF-EnSC+gLRMC [68]	λ and α similar to EWZF-SSC $\zeta \in \{0.5, 0.7, 0.9\}$

matrix completion with Grassmannian Rank-One Subspace Estimation, GROUSE [7], is performed to get a subspace estimate, and then vectors are reassigned, and the process repeated until convergence. The algorithm stops when the clusters remain unchanged in successive iterations or the algorithm reaches the maximum allowed iterations

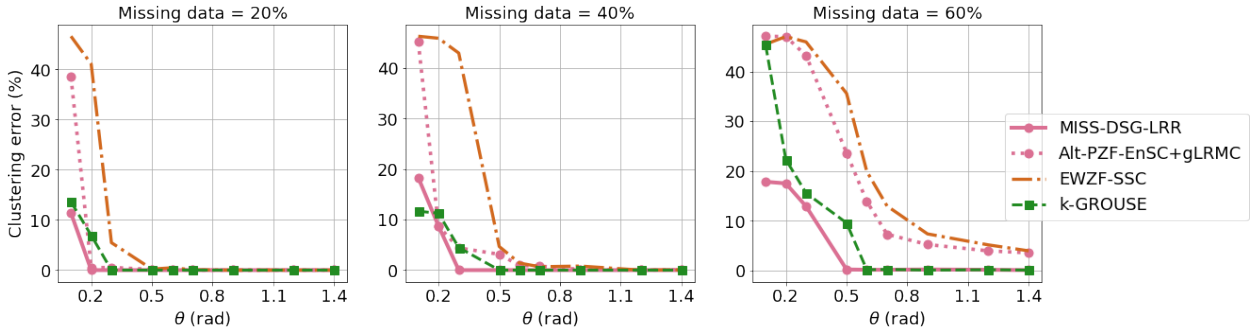
Since both MISS-DSG and MISS-DSG-LRR gave similar performance on the considered instances, we report only MISS-DSG-LRR when benchmarking against state-of-the-art methods. The best parameter configurations are selected based on the average completion error on a hold out set. As noted by [68], this approach translates more easily into practice compared to the more common approach in literature where parameter with least classification error is selected. Being an unsupervised learning task, no true cluster labels are available in practice. However, one can always hold out some observed entries as a validation set. In our experiments, we hold out 25% of the data in validation set for parameter selection. We report parameter choices for different algorithms in Table 3.5.

Independent subspaces: We keep the same experimental setting for independent subspaces as in Section 3.5.3. We report these results in Figure 3.6. As expected, EWZF-SSC fails when 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 MISS-DSG-LRR doing slightly better than k-GROUSE for $f = 60\%$. A similar trend is observed in completion errors as shown in Figure 3.6b.

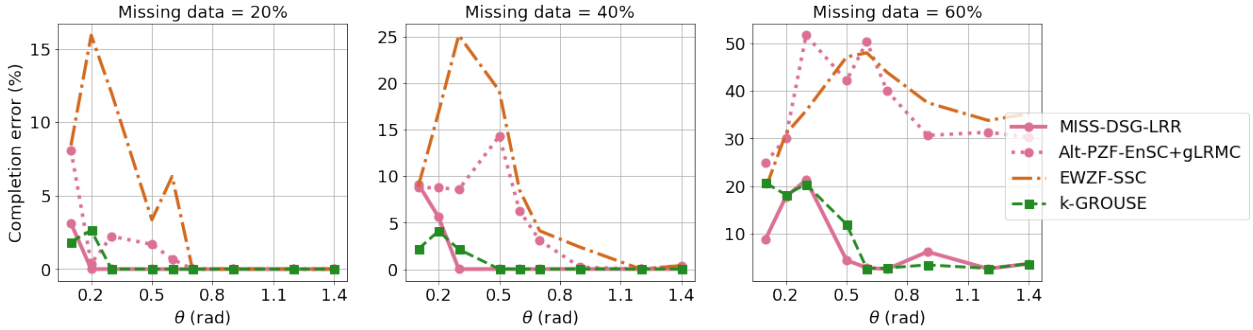
Disjoint subspaces: We fix parameters $d = 20$, $n = 200$, and $r_k = 2 \; \forall \; k \in \{1, 2, 3\}$. We vary θ between 0.2 ($\approx 12^\circ$) and 1.2 ($\approx 68^\circ$). For each value of θ, we consider 10 random trials. We let missing data percentage $f \in \{20, 40, 60\}$. All methods are provided the true number of subspaces K and the true dimension of subspaces. We report these results in Figure 3.7. We see a significant drop in performance when compared to independent subspaces for all algorithms but MISS-DSG-LRR. EWZF-SSC and Alt-PZF-EnSC+gLRMC

Figure 3.6: Performance comparison against state-of-the-art as a function of subspace angles for two independent subspaces. Parameters are $d = 20, n = 200, K = 2$, and $r_1 = r_2 = 2$

(a) Clustering error



(b) Completion error



give high clustering errors when any pair of subspaces are close to each other (small θ) or there is high amount of missing data, and are outperformed by both k-GROUSE and MISS-DSG-LRR. Performance of k-GROUSE deteriorates in low-affinity and high missing data regime. MISS-DSG-LRR is the only algorithm which gives perfect recovery of clusters, in terms of low clustering errors as well as low completion errors, in low-affinity and high missing data regime.

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 missing data between 0 to 65% as shown in Figure 3.8a. 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%), only 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 reconstruction error for $f > 30\%$ as shown in

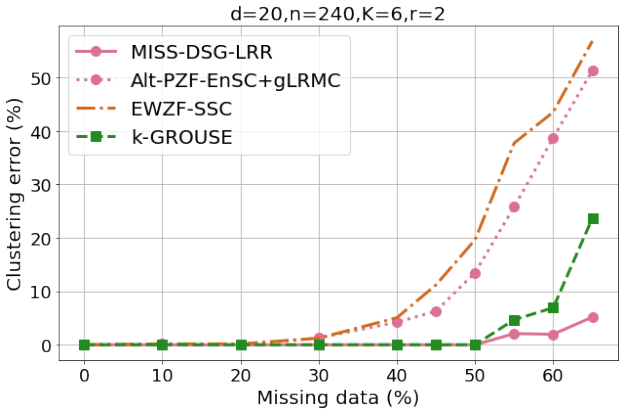
Figure 3.7: Performance comparison against state-of-the-art as a function of subspace angles for three disjoint subspaces. Parameters are $d = 20, n = 200, K = 3$, and $r_1 = r_2 = r_3 = 2$

Table 3.6. MISS-DSG-LRR gives lower completion error than k-GROUSE for $f > 50\%$ while both give no completion error for $f < 50\%$. Again ,high completion errors for MISS-DSG even after getting the right clusters for $f \in \{60, 65\}$ is attributed to the fact that low-rank matrix completion fails in that regime.

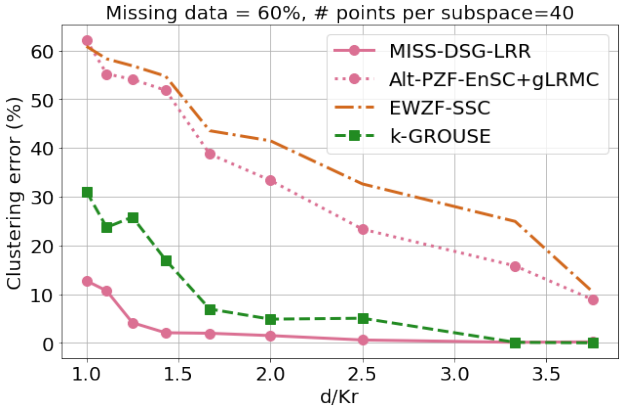
We next study the effect of total rank $(\sum_{i=1}^{K} r_i)$ with respect to the ambient dimension d on clustering error. We consider the same instances as we did in Section 3.5.3 for randomly sampled subspaces and vary $d/Kr \in [1, 4]$ as shown in Figure 3.8b. Since self-expressive methods do not 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 high-rank to low-rank regime. In the high rank ratio 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 matrix is nearly full-rank $(d/Kr < 2)$, all methods give high completion errors including MISS-DSG-LRR which has small clustering errors as shown in Table 3.7. This is due to the fact that low-rank matrix completion fails in such a setting.

Figure 3.8: 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



3.5.5 Choice of penalty parameter in MISS-DSG

If we know number of subspaces (K) and their underlying dimension $r_i \; \forall i = 1, \ldots, K$, then we can directly solve (3.10) with $(\lambda) = 0$. In many cases, we do not have that information and the choice of penalty parameter becomes crucial. A small value of λ would encourage model to use higher complexity union of subspaces model, $\mathcal{U} = \{U_1, U_2, \ldots, U_{K'}\}$. This could either mean subspaces being selected of higher dimension than ground truth and/or selecting larger number of subspaces than ground truth, i.e., $K' > K$. Similarly, a large value of λ would encourage model to use lower complexity union of subspace model, i.e., subspaces being selected of lower dimension than ground truth and/or selecting fewer subspaces than ground truth, $K' < K$. To investigate this more, we fix $d = 30, n = 300, K = 6, r = r_k = 3 \; \forall k \in [K]$, and vary $f \in \{10, 30, 50\}$. We consider three different cases for MISS-DSG:

Table 3.6: Average completion error (%) for random instances in Figure 3.8a

f	EWZF-SSC	Alt-PZF-EnSC+gLRMC	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 3.7: Average completion error (%) for random instances in Figure 3.8b

d/Kr	EWZF-SSC	Alt-PZF-EnSC+gLRMC	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

- K known, r unknown: In this case, we do not provide MISS-DSG with information on dimension of underlying subspaces. Instead, we use $r_{\max} = 2 * r = 6$ in Algorithm 4. 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 (3.14c).

- K unknown, r known: In this case, we assume that we know dimension of underlying subspaces. Hence, MISS-DSG considers subspaces only of dimension 3. Since we don't know number of subspaces (K), we remove constraint (3.14c) from our model, and let MISS-DSG self-determine the number of subspaces.

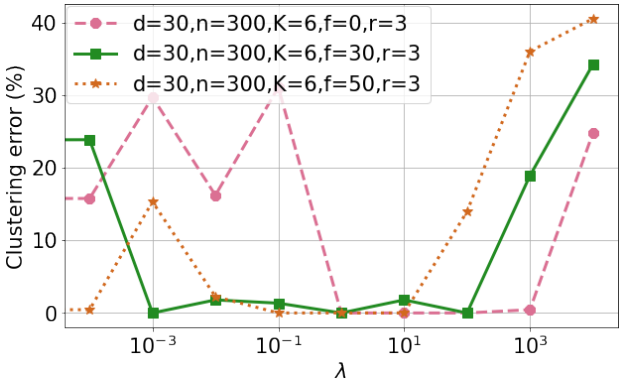
- K unknown, r unknown: Now we consider the case when we don't know dimensions as well as number of underlying subspaces. We again use $r_{\max} = 2 * r = 6$ in Algorithm 4, and hence MISS-DSG considers subspaces of dimension $\in \{1, 2, 3, 4, 5, 6\}$. Similar to previous case, we don't know K and hence constraint (3.14c) 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 3.9 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 3.9a, 3.9b). The performance is marginally better

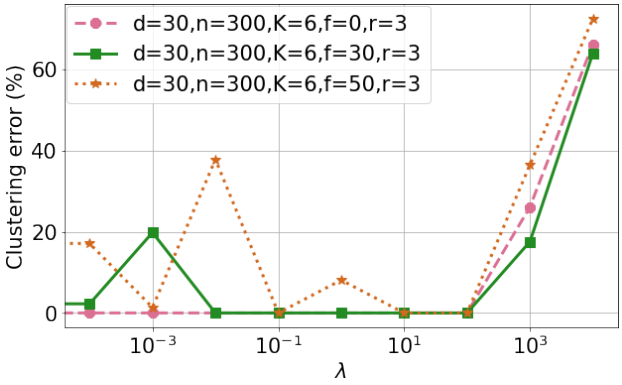
with knowledge on dimensions of subspaces than number of subspaces. Extremely high values of λ lead to high clustering errors in all cases since model is forced to either select fewer subspaces or select subspaces of lower dimension than ground truth. For $\lambda = 10^4$, MISS-DSG selected subspaces 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. We also point out that choice of λ can also vary with percentage of missing data, f. We now consider the case when we do not know both r and K as shown in Figure 3.9c. 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 ($\sum_{j\in[n]} w_j$) but have a different complexity than ground truth. Hence, choice of λ is critical in this case, and thus MISS-DSG gives low clustering error only for a narrow range of λ.

Figure 3.9: Effect of λ (x axis, log scale) on clustering error

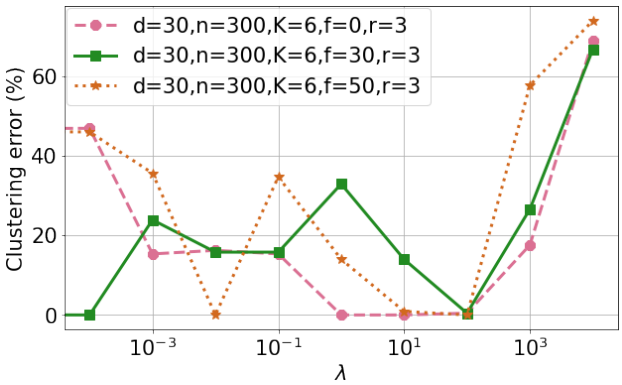
(a) K known, r unknown



(b) K unknown, r known



(c) K unknown, r unknown



3.5.6 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 [112] and demonstrate the competitive performance of MISS-DSG. We evaluate different methods on the Hopkins 155 data set 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 [39]. To simulate high rank case, similar to [122], 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 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 3.8 and MISS-DSG with initialization from Alt-PZF-EnSC+gLRMC referred to as MISS-DSG-A in Table 3.8. 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 3.5.4.

We report average clustering error over 155 sequences for each method for different missing data percentage in Table 3.8. 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 (referred to as MISS-DSG-A) 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 3.8: Performance comparison on Hopkins 155 dataset with # frames=6

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

3.5.7 Computational times

We now discuss the computational performance of MISS-DSG. We first point out that most existing methods in literature were found to be faster than MISS-DSG. Hence, we only discuss computational times of existing methods briefly and give a detailed breakup of solution time for MISS-DSG.

We first discuss the computational times of existing MILP 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 cheap (< 2 minutes for synthetic instances considered in this chapter). In fact, most 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 alone took between $4 - 25$ minutes on average for synthetic instances considered in this work. The subspace generation time varies based on the amount of missing data since LRMC also becomes expensive in presence of high-missing data. Due to a large number of candidate subspaces, solving MILP model is also computationally more expensive than BB-LRR and MB-FLOSS, and took between $2 - 10$ minutes.

EWZF-SSC on the considered synthetic instances was found to be computationally efficient and took a maximum of 2 minutes. k-GROUSE which does not require any parameter tuning took a maximum of 12 minutes. Alt-PZF-EnSC+gLRMC which combines SSC with LRMC is relatively expensive than these two methods with computational time varying between $3-120$ minutes. This also includes the parameter training time considered in Table 3.5. 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 took between $3 - 75$ minutes for the synthetic instances depending on the hardness of the instance. In particular, high missing data regime and high-rank matrix led to higher computational times. Compared to random instances, independent and disjoint instances were significantly faster and took a maximum of 5 minutes. We give a detailed breakdown of the computation time spent in each component of the MILP approach in Table 3.9. We also report number of Benders cuts generated at root node and during branching. Number of new columns added in Algorithm 4 are also reported. We observe that as missing data increases, total solution time also increases. Similarly, an increase in number of subspaces leads to increase in computational time. This is due to the fact that a higher number of columns generation iterations are performed on harder instances which results in more time being spent on solving the pricing problem. This consecutively leads to calculating more d_{jt} coefficients and regenerating new Benders cuts as new columns

Table 3.9: Detailed computational times for MILP for randomly generated instances

	# New cols	# Benders cuts		Total time (s)	Time (s)			
		Root node	Branching	(Alg. 4)	Root node Benders (Alg. 4, lines 7-15)	Gradient (Alg. 3, line 7)	Residuals calculation (Alg. 4, line 2+Alg. 3, line 3)	MIP (Alg. 4, line 30)
$\mathbf{d = 20, n = 240, K = 6, r = 2}$								
f=10	5727	6416	907	193.2	17.7	60.5	76.2	33.8
f=20	6464	7025	1031	698.8	53.3	198.3	289.6	143.6
f=40	9956	8145	870	1074.8	83.4	295.4	437.6	237.2
f=65	21744	10701	996	2661.9	287.6	759.4	902.7	663.9
$\mathbf{d \in \{20, 30\}, n = 40K, f = 60, r = 2}$								
d=20, K=3, r=2	9086	2081	381	354.2	21.3	113.5	131.6	82.0
d=30, K=4, r=2	13798	5275	608	628.7	42.6	187.4	244.8	131.5
d=20, K=5, r=2	16117	6182	592	853.7	78.6	262.3	320.2	176.9
d=20, K=7, r=2	19765	10788	870	1606.0	146.4	463.3	599.6	365.1
d=20, K=9, r=2	23281	19377	2240	4929.4	541.0	1240.0	1242.1	1811.4

are added, leading to higher computational cost. We observe from Table 3.9 that a large proportion of time ($\approx 70\%$) is spent on calculating the gradient (∇g) and computing d_{jt} coefficients. In particular, around 25-30% of the time is spent on solving the pricing problem and about 40% on computing the d_{jt} coefficients. Both of these operations 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 MISS-DSG 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 [87].

3.6 Conclusions and future directions

We proposed a novel MILP framework MISS-DSG for the subspace clustering with missing data and showed its effectiveness relative to other state-of-the-art methods, especially in certain instance regimes. 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 initial 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 but a parallel implementation can offer significant speedups. A possible future direction is to generalize this framework to a union of models (e.g, clustering mixture of Gaussian models) instead of limiting to a union of linear subspaces.

4 INTEGER PROGRAMMING APPROACHES TO BINARY MATRIX COMPLETION

4.1 Introduction

In Chapter 3, we focused on the union of subspaces model in $\mathbb{R}^d$ which has applications in various machine learning and computer vision applications. However, in many applications data is categorical (binary) rather than continuous. With binary data, it becomes useful to explore matrix completion and the union of subspace models in the case when the matrix data is binary.

In this chapter, we derive integer programming formulations for matrix factorization and completion in $\mathbb{F}_2$, the finite field of 2 elements. We first derive a natural integer programming formulation that uses general integer variables to model the parity conditions of arithmetic in $\mathbb{F}_2$. We then derive different formulations that use only binary decision variables. Our key contributions are a new class of inequalities to model the dot product (for a single matrix element), and showing that these inequalities characterize the convex hull of dot product of two vectors in $\mathbb{F}_2$. Furthermore, we derive new classes of valid inequalities linking dot products of two different matrix elements.

Matrix factorization and completion over $\mathbb{F}_2$

Low-rank matrix factorization is a classical problem where the goal is to approximate a given matrix X of size $d \times n$ as the product of two matrices U of size $d \times r$ and V of size $r \times n$ such that $X \approx UV$. Columns of U are interpreted as the factors or basis vectors of the low-dimensional subspace of rank r, and each column of V contains the combination coefficients [33]. Based on the application, different restrictions are often imposed on factor matrices U, V, e.g., orthogonality [51], non-negativity [73], and others. The problem is well studied when matrices are defined over $\mathbb{R}$, where low-rank approximation can be efficiently solved via the Singular Value Decomposition [116].

Our focus is on finding a rank r approximation of binary data matrix $X \in \{0,1\}^{d\times n}$. In particular, we seek to find matrices U and V such that $X \approx UV$ with $U \in \{0,1\}^{d\times r}$ and $V \in \{0,1\}^{r\times n}$. The motivation for binary restriction is that in many machine learning applications, e.g., classification, recommendation systems, community detection, cryptography, and others, involve data with binary values, and discovering their latent structure is often useful [34]. In addition to the restriction on factor matrices U and V, one can also change the underlying arithmetic to arrive at different decompositions [34]. In this

chapter, we focus on the case when multiplication of U and V is defined with respect to $\mathbb{F}_2$ model as $X = U \otimes V$. The finite field $\mathbb{F}_2$ consists of two elements: 0 and 1, and follows modulo-2 arithmetic. The dot product of two binary vectors u, v in $\mathbb{F}_2$ model is defined as $u^T v := \bigoplus_i u_i v_i$, where $\oplus$ denotes modulo-2 addition. Note that the dot product over $\mathbb{F}_2$ is not an inner product and does not induce a norm: there exists $a \neq 0$ such that $a^T a = 0$ over $\mathbb{F}_2$, and hence no singular value decomposition (SVD) exists for matrices defined over $\mathbb{F}_2$ [33].

In matrix completion over $\mathbb{F}_2$, we are given a partially observed binary data matrix X_Ω, where Ω is the set of observed indices of the matrix X. Columns of X are assumed to lie in a subspace of dimension $r \ll \min\{d, n\}$, and the goal is to fill in the missing entries. Matrix factorization is a popular approach for matrix completion, since X is hypothesized to be of low rank r, and hence can be decomposed as product of two smaller matrices, $U \in \{0,1\}^{d \times r}$ and $V \in \{0,1\}^{r \times n}$. Thus, the goal of matrix completion is to find two binary matrices $U \in \{0,1\}^{d \times r}$ and $V \in \{0,1\}^{r \times n}$ such that the error on observed entries $\sum_{(ij) \in \Omega} |X_{ij} - Z_{ij}|$ is minimized, where $Z = U \otimes V$.

Notation

We use $\oplus$ to denote addition in $\mathbb{F}_2$ and $\otimes$ to denote multiplication in $\mathbb{F}_2$. We use $\otimes$ to denote dot product of two vectors in $\mathbb{F}_2$ as well as the product of two matrices in $\mathbb{F}_2$. We let e_i denote a $r-$dimensional vector of all $0'$s with 1 at i^{th} position and $\mathbf{0}$ denote a $r-$dimensional vector of all $0's$. For an integer $m \in \mathbb{Z}_+$, we use notation $[m]$ to denote the set $\{1, 2, \ldots, m\}$. For a vector $c \in R^r$ and set $S \subseteq [r]$, the notation $c_S = a$ represents $c_l = a \; \forall \; l \in S$, and $c(S)$ represents the sum $\sum_{l \in S} c_l$.

Literature survey on low-rank binary matrix completion

We first note that using classical methods, such as SVD, to attain a low-rank factorization of a binary matrix X is unlikely to yield binary matrices U and V. Furthermore, simply imposing binary constraints on matrices U and V is insufficient either since the standard integer product would result in non-binary values as shown in an example below:

$$U = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, V = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, Z = UV = \begin{bmatrix} 1 & 0 \\ 2 & 0 \end{bmatrix}.$$

Ruling out the standard integer product, there are two natural formulations depending on the definition of the vector dot product. One formulation uses Boolean arithmetic where the product of the binary matrices U and V is computed by interpreting 0s as false and 1s

as true, using logical disjunction ($\vee$) in place of addition, and using logical conjunction ($\wedge$) in place of multiplication. The Boolean matrix product is then

$$Z = U \circ V \iff z_{ij} = \bigvee_{k \in [r]} (u_{ik} \wedge v_{kj}).$$

Over the last decade, there has been continuous interest in Boolean matrix factorization. We refer reader to [85] for a recent survey on Boolean matrix factorization. Recently, integer programming based methods for Boolean matrix factorization and completion have also been explored [65, 85].

A second formulation is based on the Galois field $\mathbb{F}_2$ which follows modulo-2 arithmetic. Our focus in this chapter is on $\mathbb{F}_2$ model for two reasons. First, the $\mathbb{F}_2$ model is less studied in the literature than the Boolean model. The underlying combinatorial and discrete nature of the problem naturally calls for integer programming based methods capable of exploiting the inherent problem structure. However, such a method is lacking for $\mathbb{F}_2$. And second, Boolean algebra does not define a field, and hence there is no notion of a linear space, making $\mathbb{F}_2$ a natural choice for union of subspace model for binary data for future studies.

The problem of reconstructing X when X is partially observed (X_Ω) and elements of X lie in $\mathbb{R}$ has been extensively studied [89]. The problem has received less attention for finite fields. Nonetheless, reconstructing X from X_Ω and factoring X into binary matrices has applications in network and index coding, independent component analysis, social networks, market-based data clustering, DNA transcription profiles, and others [76, 66]. We next give a brief literature review of applications and algorithms for matrix factorization and completion in $\mathbb{F}_2$ both of which are known to be *NP*-hard [54].

Index coding is the study of coding schemes to broadcast multiple messages to receivers that may have different side information [16]. A server holds a set of messages that it wishes to broadcast over a noiseless channel to a set of receivers. Each receiver is interested in one of the messages and has side-information comprising some subset of the other messages [18]. Index coding has important engineering applications such as satellite communication, content broadcasting, distributed caching, device-to-device relaying, and interference management [2].

With a linear encoding scheme, all encoding operations are linear over a finite field. In a breakthrough paper, Bar-Yossef et al. [11] showed the equivalence between linear index coding and rank minimization over a finite field. Following this, a number of heuristic methods in the index coding literature were developed. Esfahanizadeh et al. [40] proposed a matrix completion algorithm for the linear index coding problem. In their algorithm, the complete sub-matrix of highest rank is identified using a heuristic scheme which is

then expanded using row and column projections such that the possible completions of an incomplete row or column are in the span of the complete sub-matrix. Tan et al. [110] studied the rank minimization problem where, instead of entries, random linear combinations of the entries are observed. They give various information-theoretic bounds on the number of measurements necessary for low-rank matrix recovery.

Dan et al. [33] considered the problem of rank r binary matrix factorization from the lens of column subset selection (CSS), in which one low rank matrix must be formed by r columns of the data matrix. They provided a tight upper bound on the approximation ratio of CSS for binary matrices over $\mathbb{F}_2$. Wicker et al. [120] provided a heuristic factorization algorithm where they construct rank-1 factor matrices iteratively with one of the matrix being forced to use rows from the data matrix X, something similar to CSS, but instead applied on rows.

Fomin et al. [47] and Ban et al. [10] independently defined a polynomial-time approximation scheme for low-rank approximation of binary matrices over $\mathbb{F}_2$ as well as in Boolean algebra. They analyzed the approximation algorithm for the generalized clustering algorithm which encompassed low-rank approximation over $\mathbb{F}_2$ and like Dan et al. [33] assumed that the matrix is fully observed. Both Fomin et al. [47] and Ban et al. [10] provided factorization algorithms for the best rank r approximation over $\mathbb{F}_2$ with $(1+\epsilon)$-approximations. As pointed out by Fomin et al. [46], the algorithms developed in [47, 10] are good for theoretical analysis but rather impractical due to tremendous running times. Kumar et al. [66] then gave the fastest known bicriteria constant-factor approximation algorithm for the rank r factorization over $\mathbb{F}_2$ when the matrix is fully observed.

The work most closely related to ours is of Saunderson et al. [106] who proposed a linear programming based algorithm for matrix completion over $\mathbb{F}_2$. They infer linear relations among the rows and columns of X_Ω, and construct collections of parity check sets for rows (H_1) and columns (H_2). The collections of parity checks are then used to construct row and column bases. For $S \subseteq [d]$, $e_S \in \mathbb{F}_2^d$ is the vector supported on S, i.e. $[e_S]_i = 1$ if and only if $i \in S$ and 0 otherwise. They construct H_1 by ensuring *apparent* consistency, i.e,

$$\left\{ Y \in {\mathbb{F}_2}^{d \times n} : Y_\Omega = X_\Omega, e_{S_1}^T \otimes X = 0 \; \forall \; S_1 \in H_1 \right\} \neq \emptyset.$$

In other words, they consider a subset of rows that sum to 0 and then use this information to construct a basis whose columns span $\{e_{S_i} : S_i \in H_1\}^\perp$. H_2 is constructed similarly. They established conditions on X and Ω when the collections of parity checks are consistent. After generating H_1 and H_2 in polynomial time, their proposed algorithms selects subsets of H_1 and H_2. They formulate this subset selection problem as a combinatorial optimization

problem with the objective of maximizing the number of parity checks kept and solved the LP relaxation of the same. Unfortunately, the proposed algorithm is not exact.

Although there have been some theoretical advances on matrix factorization over $\mathbb{F}_2$, practical algorithms are still lacking. Some heuristic approaches have been proposed for factorization in recent years as discussed above but an exact algorithm is yet to be developed. Moreover, excluding the approximation algorithm of [106], irrespective of the underlying combinatorial and discrete nature of the problem, no linear or integer programming based methods have been studied. The factorization problem in $\mathbb{F}_2$ with missing data has not been studied either.

Contributions

We derive integer linear programming formulations for the matrix factorization and completion models over $\mathbb{F}_2$. We first derive formulations making use of McCormick envelopes for product of two binary variables: a base formulation using an integer variable and an extended formulation using ideas from disjunctive programming and parity polytopes. The latter formulation characterizes the convex hull of the dot product of two vectors in an extended space. We then derive a novel formulation based on a new class of valid inequalities that also characterizes the convex hull of the dot product of two vectors in the original space of variables. Furthermore, we derive new classes of valid inequalities linking dot products between two matrix elements.

4.2 MILP for matrix factorization in $\mathbb{F}_2$

We wish to model $Z = U \otimes V$ in $\mathbb{F}_2$, where U and V are matrices to be determined with elements denoted by $u_{ik}, v_{kj}\ \forall i \in [d], j \in [n], k \in [r]$. We let z_{ij} denote the dot product of u_i and v_j in $\mathbb{F}_2$, i.e, $z_{ij} = u_i \otimes v_j$. Consecutively, $Z = U \otimes V$. Let $\Omega \subseteq [d] \times [n]$ denote the set of observed indices of binary matrix X. The goal is to find binary matrices U and V such that the error on observed entries $\sum_{(i,j)\in\Omega} |X_{ij} - Z_{ij}|$ is minimized while ensuring $Z = U \otimes V$. When X is fully-observed, the objective minimizes the error on all entries of X, and for an incomplete X with missing entries, the objective minimizes error on the observed entries.

We wish to solve the following optimization problem:

$$\min \sum_{(i,j)\in\Omega} \big[X_{ij}(1-z_{ij}) + (1-X_{ij})z_{ij}\big] \tag{4.1a}$$

$$z_{ij} = \bigoplus_{k\in[r]} (u_{ik}v_{kj}) \quad \forall i \in [d], j \in [n], \tag{4.1b}$$

$$u \in \{0,1\}^{d\times r}, v \in \{0,1\}^{r\times n}. \tag{4.1c}$$

It is easy to see that the objective of formulation (4.1) is the same as $\sum_{(i,j)\in\Omega} |X_{ij} - Z_{ij}|$. We point out that constraint (4.1b) in formulation (4.1) is the key constraint that encodes the $\mathbb{F}_2$ arithmetic. Since constraint (4.1b) is the same for each (i,j) pair, we construct the integer linear programming by modeling the constraint (4.1b) for a fixed (i,j), which is then repeated for all $(i,j) \in [d] \times [n]$.

We let y_{ijk} denote the product of u_{ik} and v_{kj}, i.e., $y_{ijk} = u_{ik}v_{kj}$. Then, z_{ij} can equivalently be written as $z_{ij} = \bigoplus_{k\in[r]} y_{ijk}$. We let $y_{ij} \in \{0,1\}^r$ denote the vector $[y_{ij1}, y_{ij2}, \ldots, y_{ijr}]$.

For each $(i,j) \in [d] \times [n]$, we define the set $\mathcal{I}_{ij}$ to denote the dot product of u_i, v_j in $\mathbb{F}_2$, set $\mathcal{P}_{ij}$ to denote addition in $\mathbb{F}_2$, and the set $\mathcal{M}^I_{ij}$ to denote scalar product. Specifically, we define the following three sets.

Definition 4.1. $\mathcal{I}_{ij} := \{(u_i, v_j, z_{ij}) \in \{0,1\}^r \times \{0,1\}^r \times \{0,1\} \mid z_{ij} = \bigoplus_{k\in[r]} u_{ik}v_{kj}\}$

Definition 4.2. $\mathcal{P}_{ij} := \{(u_i, v_j, y_{ij}, z_{ij}) \in \mathbb{R}^r \times \mathbb{R}^r \times \{0,1\}^r \times \{0,1\} \mid z_{ij} = \bigoplus_{k\in[r]} y_{ijk}\}$

Definition 4.3. $\mathcal{M}^I_{ij} := \{(u_i, v_j, y_{ij}, z_{ij}) \in \{0,1\}^r \times \{0,1\}^r \times \{0,1\}^r \times \mathbb{R} \mid y_{ijk} = u_{ik}v_{kj}\ \forall k \in [r]\}$

It is well known that constraint $y_{ijk} = u_{ik}v_{kj}$ in the definition of $\mathcal{M}^I_{ij}$ is interchangeable with the McCormick envelope linear constraints: $y_{ijk} \leq u_{ik}, y_{ijk} \leq v_{kj}$, and $y_{ijk} \geq u_{ik} + v_{kj} - 1$ [84].

With these definitions, the optimization problem (4.1) can be re-written as follows:

$$\min \sum_{(i,j)\in\Omega} \big[X_{ij}(1-z_{ij}) + (1-X_{ij})z_{ij}\big] \tag{4.2a}$$

$$(u_i, v_j, z_{ij}) \in \mathcal{I}_{ij} \quad \forall i \in [d], j \in [n]. \tag{4.2b}$$

Since sets $\mathcal{I}_{ij}, \mathcal{P}_{ij}$, and $\mathcal{M}^I_{ij}$ are defined for all $(i,j) \in [d] \times [n]$, for ease of notation, we drop

the subscript (ij) and define sets $\mathcal{I}, \mathcal{P}$, and $\mathcal{M}^I$ as follows:

$$\mathcal{I} := \{(u, v, z) \in \{0,1\}^r \times \{0,1\}^r \times \{0,1\} \mid z = \bigoplus_{k\in[r]} u_k v_k\}, \tag{4.3a}$$

$$\mathcal{P} := \{(u, v, y, z) \in \mathbb{R}^r \times \mathbb{R}^r \times \{0,1\}^r \times \{0,1\} \mid z = \bigoplus_{k\in[r]} y_k\}, \tag{4.3b}$$

$$\mathcal{M}^I := \{(u, v, y, z) \in \{0,1\}^r \times \{0,1\}^r \times \{0,1\}^r \times \mathbb{R} \mid y_k = u_k v_k \; \forall k \in [r].\} \tag{4.3c}$$

Lemma 4.4. *The set $\mathcal{I}$ is the projection of the intersection of the sets $\mathcal{P}$ and $\mathcal{M}^I$, i.e.,*

$$\mathcal{I} = \text{proj}_{u,v,z}(\mathcal{P} \cap \mathcal{M}^I).$$

Proof. The proof follows by substituting y_k with $u_k v_k$ in the intersection of $\mathcal{P}$ and $\mathcal{M}^I$ as follows:

$$\begin{aligned}\text{proj}_{u,v,z}(\mathcal{P} \cap \mathcal{M}^I) &= \text{proj}_{u,v,z}(\{(u, v, y, z) \in \{0,1\}^{3r+1} \mid z = \bigoplus_{k\in[r]} y_k, y_k = u_k v_k \; \forall k \in [r]\}) \\ &= \{(u, v, z) \in \{0,1\}^{2r+1} \mid z = \bigoplus_{k\in[r]} u_k v_k\} = \mathcal{I}.\end{aligned}$$ □

Using Lemma 4.4, we can then rewrite the optimization problem (4.2) as follows:

$$\min \sum_{(i,j)\in\Omega} \left[X_{ij}(1 - z_{ij}) + (1 - X_{ij})z_{ij}\right] \tag{4.4a}$$

$$(u_i, v_j, y_{ij}, z_{ij}) \in \mathcal{M}^I_{ij} \cap \mathcal{P}_{ij} \quad \forall i \in [d], j \in [n]. \tag{4.4b}$$

Proposition 4.5. *The set $\text{conv}(\mathcal{I})$ is full-dimensional, i.e., dim$(\text{conv}(\mathcal{I}))$=$2r + 1$.*

Proof. We construct $2r + 2$ affinely independent points (u, v, z) in $\mathcal{I}$: $(\mathbf{0}, \mathbf{0}, 0), (e_1, \mathbf{0}, 0), (e_2, \mathbf{0}, 0), \ldots, (e_r, \mathbf{0}, 0), (\mathbf{0}, e_1, 0), (\mathbf{0}, e_2, 0), \ldots, (\mathbf{0}, e_r, 0), (e_1, e_1, 1)$. This implies that dimension of $\text{conv}(\mathcal{I})$ is $2r + 2 - 1$. □

We next present three different integer programming formulation for matrix factorization over $\mathbb{F}_2$. In Section 4.2.1 and 4.2.2, we derive integer programming formulations using the optimization problem (4.4), i.e., we model sets $\mathcal{M}^I_{ij}$ and $\mathcal{P}_{ij}$ separately, and the resulting formulation is in an extended space of (u, v, y, z) variables in dimension $3r + 1$. In Section 4.2.3, we derive an integer programming formulation using optimization problem (4.2), i.e., we model $\mathcal{I}_{ij}$ directly, and hence the formulation is in space of (u, v, z) variables in dimension $2r + 1$.

4.2.1 Formulation I: McCormick with general integer variable

We model the factorization problem as a quadratically constrained program and model the modulo-2 addition logic for $\mathbb{F}_2$ by introducing a new auxiliary variable:

$$\min \sum_{(i,j)\in\Omega} \big[X_{ij}(1-z_{ij}) + (1-X_{ij})z_{ij}\big] \tag{4.5a}$$

$$y_{ijk} = u_{ik}v_{kj} \qquad \forall i \in [d], j \in [n], k \in [r] \tag{4.5b}$$

$$\sum_{k\in[r]} y_{ijk} - z_{ij} - 2t_{ij} = 0 \qquad \forall i \in [d], j \in [n] \tag{4.5c}$$

$$u \in \{0,1\}^{d\times r}, v \in \{0,1\}^{r\times n}, z \in \{0,1\}^{d\times r}, t \in \mathbb{Z}_+^{d\times n}. \tag{4.5d}$$

If $\sum_{k\in[r]} y_{ijk} = 2m$ for some $m \in \mathbb{Z}_+$, then the only feasible solution to satisfy (4.5c) is $z_{ij} = 0, t = m$. If $\sum_{k\in[r]} y_{ijk} = 2m+1$ for some $m \in \mathbb{Z}_+$, then the only feasible solution to satisfy (4.5c) is $z_{ij} = 1, t = m$. Hence, constraint (4.5c) is able to model addition in $\mathbb{F}_2$. Constraints (4.5b) can equivalently be replaced with McCormick envelope representation of $\mathcal{M}_{ij}^I$ yielding the following integer linear programming formulation:

$$\min \sum_{(i,j)\in\Omega} \big[X_{ij}(1-z_{ij}) + (1-X_{ij})z_{ij}\big] \tag{4.6a, Formulation I}$$

$$y_{ijk} \geq u_{ik} + v_{kj} - 1, y_{ijk} \leq u_{ik}, y_{ijk} \leq v_{kj} \qquad \forall i \in [d], j \in [n], k \in [r] \tag{4.6b}$$

$$\sum_{k\in[r]} y_{ijk} - z_{ij} - 2t_{ij} = 0 \qquad \forall i \in [d], j \in [n] \tag{4.6c}$$

$$u \in \{0,1\}^{d\times r}, v \in \{0,1\}^{r\times n}, z \in \{0,1\}^{d\times n}, t \in \mathbb{Z}_+^{d\times n}. \tag{4.6d}$$

4.2.2 Formulation II: McCormick with parity polytopes disjunction

Our second formulation is derived by constructing an extended formulation of the parity set $\mathcal{P}_{ij}$ for each $(i,j) \in [d] \times [n]$. We use ideas of odd and even parity polytopes, along with disjunctive programming. The parity polytope is the convex hull of all $0-1$ vectors with an even number of ones [67]. A closely-related polytope is the convex hull of all $0-1$ vectors with an odd number of ones. We refer to the two polytopes as P^{even} and P^{odd} :

$$P^{\text{even}} = \text{conv}\{y \in \{0,1\}^r : \sum_{k=1}^r y_k \text{ is even }\}$$

$$P^{\text{odd}} = \text{conv}\{y \in \{0,1\}^r : \sum_{k=1}^r y_k \text{ is odd }\}.$$

We present a compact formulation in extended space, called the switching formulation, for P^{even} and P^{odd}. We state the switching formulation directly which characterizes the convex hull, and refer reader to [67] for more details. We introduce $w_j \in \mathbb{R}$ variables for $j \in \{s,t\} \cup \{[r] \times \{L,R\}\}$, i.e., $w \in \mathbb{R}^{2r+2}$. The compact formulation for P^{even} in the extended space is as follows:

$$
\begin{aligned}
P_s^{\text{even}} := \Big\{ (y,w) \in [0,1]^r \times \mathbb{R}^{2r+2} \,|\, & w_s - w_t \geq 1 && && \text{(4.7a)}\\
& w_{1,R} - w_s - y_1 \geq -1 && && \text{(4.7b)}\\
& w_{1,L} - w_s + y_1 \geq 0 && && \text{(4.7c)}\\
& w_{k+1,L} - w_{k,R} - y_{k+1} \geq -1 && \forall k \in [r-1] && \text{(4.7d)}\\
& w_{k+1,R} - w_{k,R} + y_{k+1} \geq 0 && \forall k \in [r-1] && \text{(4.7e)}\\
& w_{k+1,R} - w_{k,L} - y_{k+1} \geq -1 && \forall k \in [r-1] && \text{(4.7f)}\\
& w_{k+1,L} - w_{k,L} + y_{k+1} \geq 0 && \forall k \in [r-1] && \text{(4.7g)}\\
& w_t - w_{n,R} \geq 0 \Big\}. && && \text{(4.7h)}
\end{aligned}
$$

The switching formulation P_s^{odd} for P^{odd} is identical to P_s^{even} with the constraint (4.7h) being replaced with $w_t - w_{n,L} \geq 0$.

The disjunction of P^{odd} and P^{even} can be used to model addition in $\mathbb{F}_2$. In particular, for each $(i,j) \in [d] \times [n]$, we require z_{ij} to take a value of 1 if $\sum_{k\in[r]} y_{ijk}$ is odd and 0 if $\sum_{k\in[r]} y_{ijk}$ is even. In other words, either $y_{ij} \in P^{\text{odd}}$ or $y_{ij} \in P^{\text{even}}$. For each $(i,j) \in [d] \times [n]$, we define variables $y^o_{ij} \in \{0,1\}^r$ such that $y^o_{ij} \in P^{\text{odd}}$ if $z_{ij} = 1$, and similarly we define $y^e_{ij} \in \{0,1\}^r$ such that $y^e_{ij} \in P^{\text{even}}$ if $z_{ij} = 0$. We now enforce the constraints $y_{ijk} = y^o_{ijk} + y^e_{ijk}$, $y^o_{ijk} \leq z_{ij}$, and $y^e_{ijk} \leq 1 - z_{ij}$ for all $k \in [r]$. The latter constraints imply that if $z_{ij} = 1$, then $y_{ij} = y^o_{ij}$, and hence $y_{ij} \in P^{\text{odd}}$. Similarly, if $z_{ij} = 0$, then $y_{ij} = y^e_{ij}$, and hence $y_{ij} \in P^{\text{even}}$.

For each matrix element $(i,j) \in [d] \times [n]$, we define the set $\mathcal{D}^I_{ij}$ to be the set of points satisfying the disjunction of polytopes P^{odd} and P^{even}. Since z_{ij} already indicates which set among P^{even} and P^{odd} is selected, we don't need an auxiliary binary variable to model the disjunction between membership in the odd or even parity polytope. We use the switching formulations P_s^{odd} and P_s^{even} to represent odd and even parity polytopes. For

ease of notation, we drop index (ij), and define $\mathcal{D}^I$ as follows:

$$\mathcal{D}^I = \Big\{(w^o, w^e, y, y^o, y^e, z) \in \mathbb{R}^{2r+2} \times \mathbb{R}^{2r+2} \times \{0,1\}^r \times \{0,1\}^r \times \{0,1\}^r \times \{0,1\} \mid$$

$$w^o_s - w^o_t \geq z \tag{4.8a}$$
$$w^o_{1,R} - w^o_s - y^o_1 \geq -z \tag{4.8b}$$
$$w^o_{1,L} - w^o_s + y^o_1 \geq 0 \tag{4.8c}$$
$$w^o_{k+1,L} - w^o_{k,R} - y^o_{k+1} \geq -z \qquad \forall k \in [r-1] \tag{4.8d}$$
$$w^o_{k+1,R} - w^o_{k,R} + y^o_{k+1} \geq 0 \qquad \forall k \in [r-1] \tag{4.8e}$$
$$w^o_{k+1,R} - w^o_{k,L} - y^o_{k+1} \geq -z \qquad \forall k \in [r-1] \tag{4.8f}$$
$$w^o_{k+1,L} - w^o_{k,L} + y^o_{k+1} \geq 0 \qquad \forall k \in [r-1] \tag{4.8g}$$
$$w^o_t - w^o_{n,L} \geq 0 \tag{4.8h}$$
$$w^e_s - w^e_t \geq 1 - z \tag{4.8i}$$
$$w^e_{1,R} - w^e_s - y^e_1 \geq -(1-z) \tag{4.8j}$$
$$w^e_{1,L} - w^e_s + y^e_1 \geq 0 \tag{4.8k}$$
$$w^e_{k+1,L} - w^e_{k,R} - y^e_{k+1} \geq -(1-z) \qquad \forall k \in [r-1] \tag{4.8l}$$
$$w^e_{k+1,R} - w^e_{k,R} + y^o_{k+1} \geq 0 \qquad \forall k \in [r-1] \tag{4.8m}$$
$$w^e_{k+1,R} - w^e_{k,L} - y^e_{k+1} \geq -(1-z) \qquad \forall k \in [r-1] \tag{4.8n}$$
$$w^e_{k+1,L} - w^e_{k,L} + y^e_{k+1} \geq 0 \qquad \forall k \in [r-1] \tag{4.8o}$$
$$w^e_t - w^e_{n,R} \geq 0 \tag{4.8p}$$
$$y_k = y^o_k + y^e_k \qquad \forall k \in [r] \tag{4.8q}$$
$$y^o_k \leq z \qquad \forall k \in [r] \tag{4.8r}$$
$$y^e_k \leq 1 - z \qquad \forall k \in [r]\Big\}. \tag{4.8s}$$

Constraints (4.8a)-(4.8h) correspond to P^{odd} and constraints (4.8i)-(4.8p) correspond to P^{even}. Constraints (4.8q)-(4.8s) ensure that $y = y^o$ and $y^e = \mathbf{0}$ if $z = 1$, i.e., P^{odd}_s constraints are active, and similarly $y = y^e$ and $y^o_k = \mathbf{0}$ if $z = 0$, i.e., P^{even}_s constraints are active. We refer to the LP relaxation of $\mathcal{D}^I$ as $\mathcal{D}$.

Theorem 4.6. *The convex hull of parity set $\mathcal{P}$ is the set of (y, z) for which there exist vectors (w^o, w^e, y^o, y^e) such that $(w^o, w^e, y, y^o, y^e, z) \in \mathcal{D}$, i.e., $\text{conv}(\text{proj}_{y,z} \mathcal{P}) = \text{proj}_{y,z} \mathcal{D}$.*

Proof. This follows from the theorem of Balas [6, 31] since P^{even} and P^{odd} are non-empty polytopes. □

We now use the idea of the disjunctive formulation and parity polytopes to replace constraint (4.6b) in Formulation I (4.6) to get the following formulation:

$$\min \sum_{(i,j)\in\Omega} \left[X_{ij}(1-z_{ij}) + (1-X_{ij})z_{ij}\right] \qquad \text{(4.9a, Formulation II)}$$

$$y_{ijk} \geq u_{ik} + v_{kj} - 1, y_{ijk} \leq u_{ik}, y_{ijk} \leq v_{kj} \quad \forall i \in [d], j \in [n], k \in [r] \qquad \text{(4.9b)}$$

$$(y_{ij}, y^o_{ij}, y^e_{ij}, z_{ij}) \in \mathcal{D}^I_{ij} \qquad \forall i \in [d], j \in [n]. \qquad \text{(4.9c)}$$

The LP relaxation of Formulation II characterizes the convex hull for parity sum of each matrix element in lifted space as we show in Theorem 4.7. We first define set $\mathcal{D}'$ as follows.

$$\mathcal{D}' := \{(u, v, y, z) \in \mathbb{R}^r \times \mathbb{R}^r \times [0,1]^r \times [0,1] \mid (y, z) \in \text{proj}_{y,z}\, \mathcal{D}\} \qquad (4.10)$$

Theorem 4.7. *The convex hull of intersection of sets $\mathcal{P}$ and $\mathcal{M}^I$ is $\mathcal{D}' \cap \mathcal{M}$, i.e.,*

$$\text{conv}(\mathcal{P} \cap \mathcal{M}^I) = \mathcal{D}' \cap \mathcal{M}.$$

The proof of Theorem 4.7 is based on the following intermediate results.

Lemma 4.8. *For $\mathcal{P}$ and $\mathcal{M}^I$, $\text{conv}(\mathcal{P} \cap \mathcal{M}^I) = \text{conv}(\mathcal{P}) \cap \text{conv}(\mathcal{M}^I)$*

Proof. $\text{conv}(\mathcal{P} \cap \mathcal{M}^I) \subseteq \text{conv}(\mathcal{P}) \cap \text{conv}(\mathcal{M}^I)$ trivially since $\text{conv}(\mathcal{P} \cap \mathcal{M}^I)$ is the smallest convex set containing all the points in $\mathcal{P} \cap \mathcal{M}^I$, it must be a subset of $\text{conv}(\mathcal{P})$ and $\text{conv}(\mathcal{M}^I)$ individually.

If $(\hat{u}, \hat{v}, \hat{y}, \hat{z}) \in \text{conv}(\mathcal{P}) \cap \text{conv}(\mathcal{M}^I)$, then $(\hat{u}, \hat{v}, \hat{y}, \hat{z}) = \sum_i \lambda_i (u, v, y, \hat{z})_i$ such that $1^\top \lambda = 1, \lambda \geq 0$, and $(u, v, y, \hat{z})_i \in \mathcal{M}^I \; \forall i$. Similarly $(\hat{u}, \hat{v}, \hat{y}, \hat{z}) = \sum_j \mu_j (\hat{u}, \hat{v}, y, z)_j$ such that $1^\top \mu =$

$1, \mu \geq 0$ where $(\hat{u}, \hat{v}, y, z)_j \in \mathcal{P}\ \forall j$. Substituting $\hat{u} = \sum_i \lambda_i u_i$ and $\hat{v} = \sum_i \lambda_i v_i$,

$$\begin{aligned}
(\hat{u}, \hat{v}, \hat{y}, \hat{z}) &= \sum_j \mu_j (\hat{u}, \hat{v}, y, z)_j \\
&= \sum_j \mu_j (\hat{u}, \hat{v}, y_j, z_j) \\
&= \sum_j \mu_j \Big(\sum_i \lambda_i u_i, \sum_i \lambda_i v_i, y_j, z_j \Big) \\
&= \sum_j \mu_j \Big(\sum_i \lambda_i u_i, \sum_i \lambda_i v_i, \sum_i \lambda_i y_j, \sum_i \lambda_i z_j \Big) \\
&= \sum_j \mu_j \Big(\sum_i \lambda_i (u_i, v_i, y_j, z_j) \Big) \\
&= \sum_j \sum_i \mu_j \lambda_i (u_i, v_i, y_j, z_j) \\
&= \sum_{i,j} \gamma_{i,j} (u, v, y, z)_{ij},
\end{aligned}$$

where $\gamma_{ij} = \lambda_i \mu_j$ and $(u, v, y, z)_{ij} = (u_i, v_i, y_j, z_j)$. We observe that $\gamma_{i,j} = \lambda_i \mu_j \geq 0$,

$$\sum_{i,j} \gamma_{i,j} = \sum_{i,j} \lambda_i \mu_j = \sum_i \lambda_i \sum_j \mu_j = \sum_i \lambda_i . 1 = 1,$$

and since $(u, v, y, z)_{i,j} \in \mathcal{M}^I \cap \mathcal{P}\ \ \forall (i, j)$, it follows that $(\hat{u}, \hat{v}, \hat{y}, \hat{z}) \in \text{conv}(\mathcal{P} \cap \mathcal{M}^I)$. □

We relax $\mathcal{M}^I$ using the McCormick relaxation as follows [84]:

$$\mathcal{M} = \{(u, v, y, z) \in [0, 1]^{3r} \times \mathbb{R} : y_k \leq u_k, y_k \leq v_k, y_k \geq u_k + v_k - 1\ \forall k \in [r]\}. \tag{4.11}$$

We next show that $\mathcal{M}$ characterizes $\text{conv}(\mathcal{M}^I)$.

Proposition 4.9. *McCormick LP relaxation $\mathcal{M}$ of $\mathcal{M}^I$, characterizes convex hull of $\mathcal{M}$, i.e., $\mathcal{M} = \text{conv}\,(\mathcal{M}^I)$.*

Proof. Follows from [84] since there is no interaction between product variable in different terms.

Proof of Theorem 4.7 From Lemma 4.8, $\text{conv}(\mathcal{P} \cap \mathcal{M}^I) = \text{conv}(\mathcal{P}) \cap \text{conv}(\mathcal{M}^I)$. Using Theorem 4.4 and definition of $\mathcal{D}'$ in equation (4.10), $\text{conv}(\mathcal{P}) = \mathcal{D}'$. The result then follows by using $\text{conv}(\mathcal{M}^I) = \mathcal{M}$ from Proposition 4.9.

4.2.3 Formulation III: McCormick-free compact formulation

In the previous section, we characterized the convex hull of the set $\mathcal{I}$ in an extended space of (u, v, y, z) variables using parity polytopes, disjunctive formulations, and the McCormick relaxation. We now characterize the convex hull of $\mathcal{I}$ directly in the space (u, v, z) variables. We first define set $\mathcal{T}$ to be the set of tri-partitions on $[r]$, i.e.,

$$\mathcal{T} := \{S \subseteq [r], Q \subseteq [r], T \subseteq [r] \mid \ S \cup Q \cup T = [r], S \cap Q = \emptyset, S \cap T = \emptyset, Q \cap T = \emptyset\}.$$

For each matrix element $(i, j) \in [d] \times [n]$, we define the set $\mathcal{F}^I_{ij}$ to be the set of points satisfying the new inequalities we propose along with the bound constraints. For ease of notation, we drop the index (ij), and define $\mathcal{F}^I$ as follows:

$$\begin{aligned}\mathcal{F}^I = \Big\{ & (u, v, z) \in \{0,1\}^r \times \{0,1\}^r \times \{0,1\} \mid \\ & z + \sum_{k\in S}(u_k + v_k) - \sum_{k\in Q} u_k - \sum_{k\in T} v_k \leq 2|S| \qquad \forall (S, Q, T) \in \mathcal{T}, |S| \text{ even} \qquad (4.12a) \\ & -z + \sum_{k\in S}(u_k + v_k) - \sum_{k\in Q} u_k - \sum_{k\in T} v_k \leq 2|S| - 1 \ \ \forall (S, Q, T) \in \mathcal{T}, |S| \text{ odd} \Big\}. \qquad (4.12b)\end{aligned}$$

We refer to the LP relaxation of $\mathcal{F}^I$ as $\mathcal{F}$ and to inequalities (4.12a), (4.12b) as SQT inequalities. We next show in Theorem 4.10 that the proposed inequalities correctly encode matrix multiplication in $\mathbb{F}_2$.

Theorem 4.10. *For vectors $\hat{u} \in \{0,1\}^r, \hat{v} \in \{0,1\}^r, \hat{z} \in \{0,1\}$, $(\hat{u}, \hat{v}, \hat{z}) \in \mathcal{F}^I$ if and only if $\hat{z} = \bigoplus_{k\in[r]} \hat{u}_k \hat{v}_k$.*

Proof. We first show that if $\hat{z} = \bigoplus_{k\in[r]} \hat{u}_k \hat{v}_k$, then $(\hat{u}, \hat{v}, \hat{z}) \in \mathcal{F}^I$. For a given $\hat{u}, \hat{v}$, let

$$\begin{aligned} M_1(\hat{u}, \hat{v}) &= \{k : \hat{u}_k = \hat{v}_k = 1\} \\ M_2(\hat{u}, \hat{v}) &= \{k : \hat{u}_k = 0\} \\ M_3(\hat{u}, \hat{v}) &= \{k : \hat{v}_k = 0\}. \end{aligned}$$

- If $|M_1|$ is odd, then $\sum_{k\in[r]} \hat{u}_k \hat{v}_k = 2m + 1$ for some $m \in \mathbb{Z}_+$, and $\hat{z} = \bigoplus_{k\in[r]} \hat{u}_k \hat{v}_k = 1$. To show $(\hat{u}, \hat{v}, \hat{z}) \in \mathcal{F}^I$, we need to show that $(\hat{u}, \hat{v}, \hat{z})$ does not violate constraints (4.12a) or (4.12b). For constraint (4.12a), we maximize left-hand-side of the constraint after

moving $2|S|$ on the left, i.e.,

$$p^* = \max_{(S,Q,T)\in\mathcal{T}:|S| \text{ is even}} \{\hat{z} + \sum_{k\in S}(\hat{u}_k + \hat{v}_k - 2) - \sum_{k\in Q}\hat{u}_k - \sum_{k\in T}\hat{v}_k\}.$$

Let $(\hat{S}, \hat{Q}, \hat{T})$ be the optimal solution to the above optimization problem. We claim that $\hat{S} = M_1 \setminus \{l\}$ for some $l \in M_1$, $\hat{Q} = M_2 \cup \{l\}$, and $\hat{T} = M_3$ is the optimal solution. The objective value corresponding to above solution is $\hat{p} = \hat{z} + \sum_{M_1\setminus\{l\}}(\hat{u}_k + \hat{v}_k - 2) - \sum_{M_2\cup\{l\}}\hat{u}_k - \sum_{M_3}\hat{v}_k = 1 + 0 - 1 - 0 = 0$. We claim that there is no feasible $(S, Q, T) \in \mathcal{T}$ with $|S|$ even of higher objective value. For contradiction, assume there exists $(S', Q', T') \in \mathcal{T}$ with $|S'|$ even of objective value $p' > 0$.

We first observe that $\sum_{Q'}\hat{u}_k + \sum_{T'}\hat{v}_k \geq 1$. This follows from the fact that $|M_1|$ is odd and $|S'|$ is even resulting in $|M_1 \cap \{Q' \cup T'\}| \geq 1$. Since $\hat{u}_k = \hat{v}_k = 1\ \forall k \in M_1$, it follows that $\sum_{Q'}\hat{u}_k + \sum_{T'}\hat{v}_k \geq 1$. We next observe that $\sum_{k\in S'}(\hat{u}_k + \hat{v}_k - 2) \leq 0$ since $(\hat{u}, \hat{v})$ are binary vectors. Since $\hat{z} = 1$, it follows that $p' \leq 0$, a contradiction. Hence, $\hat{p} = 0$ is the optimal value which implies that constraint (4.12a) are satisfied.

For constraint (4.12b), we maximize left-hand-side of the constraint after moving $2|S|$ on the left, i.e.,

$$\begin{aligned} p^* &= \max_{(S,Q,T)\in\mathcal{T}:|S| \text{ is odd}} \{-\hat{z} + \sum_{k\in S}(\hat{u}_k + \hat{v}_k - 2) - \sum_{k\in Q}\hat{u}_k - \sum_{k\in T}\hat{v}_k\} \\ &\leq -\hat{z}. \end{aligned}$$

The above follows from the fact that $(\hat{u}, \hat{v})$ are binary vectors, and hence $(\hat{u}_k + \hat{v}_k - 2) \leq 0$, $-\hat{u}_k \leq 0$, and $-\hat{v}_k \leq 0$. Since $\hat{z} = 1$, $p^* \leq -\hat{z} = -1$, and hence constraints (4.12b) are satisfied.

- If $|M_1|$ is even, then $\sum_{k\in[r]}\hat{u}_k\hat{v}_k = 2m$ for some $m \in \mathbb{Z}_+$, and $\hat{z} = \bigoplus_{k\in[r]}\hat{u}_k\hat{v}_k = 0$. To show $(\hat{u}, \hat{v}, \hat{z}) \in \mathcal{F}^I$, we need to show that $(\hat{u}, \hat{v}, \hat{z})$ does not violate constraints (4.12a)or (4.12b). For constraint (4.12a), we maximize left-hand-side of the constraint after moving $2|S|$ on the left, i.e.,

$$\begin{aligned} p^* &= \max_{(S,Q,T)\in\mathcal{T}:|S| \text{ is even}} \{\hat{z} + \sum_{k\in S}(\hat{u}_k + \hat{v}_k - 2) - \sum_{k\in Q}\hat{u}_k - \sum_{k\in T}\hat{v}_k\} \\ &\leq \hat{z}. \end{aligned}$$

The above follows from the fact that $(\hat{u}, \hat{v})$ are binary vectors, and hence $(\hat{u}_k + \hat{v}_k - 2) \leq 0$, $-\hat{u}_k \leq 0$, and $-\hat{v}_k \leq 0$. Since $\hat{z} = 0$, $p^* \leq \hat{z} = 0$, and hence constraints (4.12a) are

satisfied.

For constraint (4.12b), we maximize left-hand-side of the constraint after moving $2|S|$ on the left, i.e.,

$$p^* = \max_{(S,Q,T)\in\mathcal{T}:|S| \text{ is odd}} \{-\hat{z} + \sum_{k\in S}(\hat{u}_k + \hat{v}_k - 2) - \sum_{k\in Q}\hat{u}_k - \sum_{k\in T}\hat{v}_k\}.$$

Let $(\hat{S}, \hat{Q}, \hat{T})$ be the optimal solution to the above optimization problem. We first consider the case when $M_1 \neq \emptyset$. We claim that $\hat{S} = M_1 \setminus \{l\}$ for some $l \in M_1$, $\hat{Q} = M_2 \cup \{l\}$, and $\hat{T} = M_3$ is the optimal solution. The objective value corresponding to above solution is $\hat{p} = -\hat{z} + \sum_{M_1\setminus\{l\}}(\hat{u}_k + \hat{v}_k - 2) - \sum_{M2\cup\{l\}}\hat{u}_k - \sum_{M_3}\hat{v}_k = -0 + 0 - 1 - 0 = -1$. We claim that there is no feasible $(S, Q, T) \in \mathcal{T}$ with $|S|$ even of higher objective value. For contradiction, assume there exists $(S', Q', T') \in \mathcal{T}$ with $|S'|$ even of objective value $p' > -1$.

We first observe that $\sum_{Q'}\hat{u}_k + \sum_{T'}\hat{v}_k \geq 1$. This follows from the fact that $|M_1|$ is even and $|S'|$ is odd resulting in $|M_1 \cap \{Q' \cup T'\}| \geq 1$. Since $\hat{u}_k = \hat{v}_k = 1\ \forall k \in M_1$, it follows that $\sum_{Q'}\hat{u}_k + \sum_{T'}\hat{v}_k \geq 1$. We next observe that $\sum_{k\in S'}(\hat{u}_k + \hat{v}_k - 2) \leq 0$ since $(\hat{u}, \hat{v})$ are binary vectors. Since $\hat{z} = 0$, it follows that $p' \leq -1$, a contradiction. Hence, $\hat{p} = -1$ is the optimal value which implies that constraints (4.12b) are satisfied.

If $M_1 = \emptyset$, we claim that $\sum_S(\hat{u}_k + \hat{v}_k - 2) \leq -1$ for all odd $|S|$. This follows from the fact that $M_1 = \emptyset$ implying that $\nexists k$ such that $\hat{u}_k = \hat{v}_k = 1$ implying that $\hat{u}_k + \hat{v}_k \leq 1\ \forall k \in [r]$, and the claim follows. Using this and the fact that $\hat{u}_k \leq 0, \hat{v}_k \leq 0$, it follows that $p^* \leq -1$, and hence constraints (4.12b) are satisfied.

We have thus shown that $(\hat{u}, \hat{v}, \hat{z}) \in \mathcal{F}^I$ for all $(\hat{u}, \hat{v}, \hat{z}) \in \{0,1\}^{2r+1}$ if $\hat{z} = \bigoplus_{k\in[r]} \hat{u}_k\hat{v}_k$.
We next show that for $(\hat{u}, \hat{v}, \hat{z}) \in \{0,1\}^{2r+1}$, if $\hat{z} \neq \bigoplus_{k\in[r]} \hat{u}_k\hat{v}_k$, then $(\hat{u}, \hat{v}, \hat{z}) \notin \mathcal{F}^I$.

- Let $|M_1|$ be even and $\hat{z} = 1$. Then, $\hat{z} \neq \bigoplus_{k\in[r]} \hat{u}_k\hat{v}_k$. We show that this point is violated by one of the inequalities in the $\mathcal{F}^I$. For contradiction, assume there is no inequality that violates the point $(\hat{u}, \hat{v}, \hat{z})$. Since $|M_1|$ is even, consider constraint (4.12a) and let $S = M_1, Q = M_2, T = M_3$. Then,

$$\begin{aligned} \hat{z} + \sum_{k\in S}(\hat{u}_k + \hat{v}_k) - \sum_{k\in Q}\hat{u}_k - \sum_{k\in T}\hat{v}_k &\leq 2|S| \\ 1 + \sum_{M_1}(1+1) - \sum_{M_2}0 - \sum_{M_3}0 &\leq 2|M_1| \\ 1 + 2|M_1| &\leq 2|M_1|, \end{aligned}$$

which is a contradiction.

- Let $|M_1|$ be odd and $\hat{z} = 0$. Then, $\hat{z} \neq \bigoplus_{k\in[r]} \hat{u}_k\hat{v}_k$. We show that this point is violated by one of the inequalities in the $\mathcal{F}^I$. For contradiction, assume there is no inequality that violates the point $(\hat{u}, \hat{v}, \hat{z})$. Since $|M_1|$ is odd, consider constraint (4.12b) and let $S = M_1, Q = M_2, T = M_3$. Then,

$$\begin{aligned} -\hat{z} + \sum_{k\in S}(\hat{u}_k + \hat{v}_k) - \sum_{k\in Q}\hat{u}_k - \sum_{k\in T}\hat{v}_k &\leq 2|S| - 1 \\ 0 + \sum_{M_1}(1+1) - \sum_{M_2} 0 - \sum_{M_3} 0 &\leq 2|M_1| - 1 \\ 2|M_1| &\leq 2|M_1| - 1, \end{aligned}$$

  which is a contradiction.

We have thus shown that for all $(\hat{u}, \hat{v}, \hat{z}) \in \{0,1\}^{2r+1}$ if $\hat{z} \neq \bigoplus_{k\in[r]} \hat{u}_k\hat{v}_k$, then $(\hat{u}, \hat{v}, \hat{z}) \notin \mathcal{F}^I$. □

We also state the following corollary which follows immediately from Theorem 4.10.

Corollary 4.11. *For binary vector $u, v \in \{0,1\}^r, z \in \{0,1\}$, all inequalities in the description of $\mathcal{F}^I$ are necessary to model $z = u \otimes v$.*

We next provide a valid integer programming formulation for matrix completion (4.2) problem set $\mathcal{I}$ using set $\mathcal{F}^I$ as follows.

$$\min \sum_{(i,j)\in\Omega} \big[X_{ij}(1 - z_{ij}) + (1 - X_{ij})z_{ij}\big] \qquad \text{(4.13a, Formulation III)}$$

$$(u_i, v_j, z_{ij}) \in \mathcal{F}^I_{ij} \qquad \forall i \in [d], j \in [n] \qquad (4.13b)$$

We next show that the LP relaxation of $\mathcal{F}^I$ characterizes the convex hull of $\mathcal{I}$.

Theorem 4.12. *LP relaxation of $\mathcal{F}^I$ characterizes the convex hull of $\mathcal{I}$, i.e.,*

$$\text{conv}(\mathcal{I}) = \{(u, v, z) \in [0,1]^r \times [0,1]^r \times [0,1] : \text{(4.12a)},\ \text{(4.12b)}\}.$$

Proof. We prove the result by showing that optimizing any arbitrary linear objective function over the LP relaxation yields an optimal integer solution. We show this by constructing an integer primal feasible solution and then proving it is optimal by constructing a feasible solution to the dual problem with matching objective function value.

Thus, let (c, d, f) be the coefficients of arbitrary objective function and consider the primal linear program P:

$$
\begin{aligned}
p^* = \max_{u,v,z} \quad & c^T u + d^T v + fz && && \text{(4.14a, P)}\\
& z + \sum_S (u_k + v_k) - \sum_Q u_k - \sum_T v_k \le 2|S| && \forall (S,Q,T) \in \mathcal{T} : |S| \text{ is even} \\
& && && \text{(4.14b, } \pi_{SQT}\text{)}\\
& -z + \sum_S (u_k + v_k) - \sum_Q u_k - \sum_T v_k \le 2|S| - 1 && \forall (S,Q,T) \in \mathcal{T} : |S| \text{ is odd} \\
& && && \text{(4.14c, } \pi_{SQT}\text{)}\\
& u_k \le 1 && \forall k \in [r] && \text{(4.14d, } \mu_k\text{)}\\
& v_k \le 1 && \forall k \in [r] && \text{(4.14e, } \eta_k\text{)}\\
& z \le 1 && && \text{(4.14f, } \gamma\text{)}\\
& u, v, z \ge 0, &&
\end{aligned}
$$

and its dual D

$$
\begin{aligned}
d^* = \min_{\pi,\gamma,\mu,\eta} \quad & 2\sum_{SQT} |S| \pi_{SQT} - \sum_{SQT:S_{odd}} \pi_{SQT} + 1^T \mu + 1^T \eta + \gamma && && \text{(4.15a, D)}\\
& \sum_{SQT:S \ni k} \pi_{SQT} - \sum_{SQT:Q \ni k} \pi_{SQT} + \mu_k \ge c_k && \forall k \in [r] && \text{(4.15b, } u_k\text{)}\\
& \sum_{SQT:S \ni k} \pi_{SQT} - \sum_{SQT:T \ni k} \pi_{SQT} + \eta_k \ge d_k && \forall k \in [r] && \text{(4.15c, } v_k\text{)}\\
& \sum_{SQT:S \text{ is even}} \pi_{SQT} - \sum_{S:S \text{ is odd}} \pi_{SQT} + \gamma \ge f && && \text{(4.15d, } z\text{)}\\
& \pi, \gamma, \mu, \eta \ge 0. &&
\end{aligned}
$$

Without loss of generality, we let $f = +1$.
Let $C^+ = \{k : c_k \ge 0\}$ and $D^+ = \{k : d_k \ge 0\}$. Similarly, define C^- and D^-.
We initialize primal and dual solutions as follows:

- $\hat{u}_{C^+} = 1, \hat{u}_{C^-} = 0, \hat{v}_{D^+} = 1, \hat{v}_{D^-} = 0$, and $\hat{z} = 1$ if $|C^+ \cap D^+|$ is odd and 0 otherwise.
- $\hat{\mu}_{C^+} = c_{C^+}, \hat{\mu}_{C^-} = 0$, $\hat{\eta}_{D^+} = d_{D^+}, \hat{\eta}_{D^-} = 0$, $\hat{\gamma} = f = 1$ if $|C^+ \cap D^+|$ is odd, and 0 otherwise. $\hat{\pi} = 0$

We now consider the following two cases based on whether $|C^+ \cap D^+|$ is odd or even, and construct integer primal feasible and dual feasible solutions with the same objective values.

1. $|C^+ \cap D^+|$ is odd: Observe that $(\hat{u}, \hat{v}, \hat{z})$ satisfies $\mathbb{F}_2$ arithmetic ensuring primal feasibility with objective value $\hat{p} = c(C^+) + d(D^+) + 1$.
   We next check dual feasibility. Constraints (4.15b) are satisfied since $\hat{\pi} = 0$, reducing constraints (4.15b) to $\hat{\mu}_k \geq c_k$, which are satisfied by construction of $\hat{\mu}$. Constraints (4.15c) are similarly satisfied, and constraint (4.15d) is satisfied since $\hat{\gamma} = f = 1$. The dual objective evaluates to $\hat{d} = 1^T\hat{\mu} + 1^T\hat{\eta} + \hat{\gamma} = c(C^+) + d(D^+) + 1$, which is the same as $\hat{p}$.

2. $|C^+ \cap D^+|$ is even: We define Δ^* as

$$\begin{aligned}\Delta^* = \min\{ &\min\{\min\{c_k, d_k\} : k \in C^+ \cap D^+\}, && (4.16)\\ &\min\{-d_k : \quad k \in C^+ \cap D^-\},\\ &\min\{-c_k : \quad k \in C^- \cap D^+\},\\ &\min\{-c_k - d_k : \quad k \in C^- \cap D^-\},\\ &1\}, && (4.17)\end{aligned}$$

   and observe that $\Delta^* \in [0, 1]$. If $\Delta^* < 1$, we make the following changes in our primal solution: We first make $\hat{z} = 1$. Let k^* denotes the $\arg\min_k$ corresponding to Δ^*. If $k^* \in C^+ \cap D^+$, we either update $\hat{u}_{k^*} = 0$ or $\hat{v}_{k^*} = 0$ based on whether $\Delta^* = c_{k^*}$ or $\Delta^* = d_{k^*}$. If $\Delta^* = c_{k^*} = d_{k^*}$, we only update $\hat{u}_{k^*} = 0$. If $k^* \in C^+ \cap D^-$, we update $\hat{v}_{k^*} = 1$, and similarly if $k^* \in C^+ \cap D^-$, we update $\hat{u}_{k^*} = 1$. We update both $\hat{u}_{k^*} = \hat{v}_{k^*} = 1$ if $k^* \in C^- \cap D^-$.

   If $\Delta^* = 1$, we make no changes in our primal solution.

   The above updates ensure $\mathbb{F}_2$ feasibility in $(\hat{u}, \hat{v}, \hat{z})$, and hence the primal solution is feasible with objective value $\hat{p} = c(C^+) + d(D^+) + 1 - \Delta^*$.

   We next define c', d' as follows:

$$c' = \min\{-c_k : k \in C^- \cap D^-\} \qquad d' = \min\{-d_k : k \in C^- \cap D^-\}$$

   We now construct dual feasible solution. We first update $\hat{\mu}, \hat{\eta}, \hat{\gamma}$ as follows:

$$\hat{\mu}_k = \begin{cases} c_k & \text{if } k \in C^+ \cap D^- \\ c_k - \Delta^* & \text{if } k \in C^+ \cap D^+ \\ 0 & \text{o.w.} \end{cases}, \quad \eta_k = \begin{cases} d_k & \text{if } k \in C^- \cap D^+ \\ d_k - \Delta^* & \text{if } k \in C^+ \cap D^+ \\ 0 & \text{o.w.} \end{cases}, \quad \hat{\gamma} = 1 - \Delta^* \tag{4.18}$$

   We update $\hat{\pi}$ based on whether $c' + d' \geq \Delta^*$ or $c' + d' < \Delta^*$.

(a) $c' + d' \geq \Delta^*$: We do the following update in $\hat{\pi}$:

$$\hat{\pi}_{S'Q'T'} = c' \qquad \text{where } S' = C^+ \cap D^+, Q' = C^-, T' = C^+ \cap D^- \quad (4.19)$$
$$\hat{\pi}_{S'Q''T''} = \Delta^* - \min\{c', \Delta^*\} \text{ where } S' = C^+ \cap D^+, Q'' = C^- \cap D^+, T'' = D^-$$

We let all the remaining $\hat{\pi}_{SQT}$ to 0. Observe that $\hat{\pi}_{S'Q'T'} + \hat{\pi}_{S'Q''T''} = \Delta^*$, $\hat{\pi}_{S'Q'T'} \leq -c_k \; \forall k \in C^- \cap D^-$, and $\hat{\pi}_{S'Q''T''} \leq -d_k \; \forall k \in C^- \cap D^-$.

We now show that the dual assignment in (4.18), (4.19) is feasible with objective value $\hat{d} = \hat{p}$.

Constraints (4.15b):

- If $k \in C^+ \cap D^+$, then $k \in S'$, and constraints (4.15b) can be simplified to $\hat{\pi}_{S'Q'T'} + \hat{\pi}_{S'Q''T''} + \hat{\mu}_k = \Delta^* + (c_k - \Delta^*) \geq c_k$, and are clearly satisfied.
- If $k \in C^- \cap D^+$, then $k \in Q' \cap Q''$, and constraints (4.15b) can be simplified to $-\hat{\pi}_{S'Q'T'} - \hat{\pi}_{S'Q''T''} = -\Delta^* \geq c_k$. The later follows from the definition of Δ^*.
- If $k \in C^+ \cap D^-$, then $k \in T' \cap T''$, and constraints (4.15b) can be simplified to $\hat{\mu}_k \geq c_k$ and are satisfied since $\hat{\mu}_k = c_k \; \forall k \in C^+ \cap D^-$.
- If $k \in C^- \cap D^-$ then $k \in Q' \setminus Q''$ and constraints (4.15b) can be simplified to $-\hat{\pi}_{S'Q'T'} \geq c_k$ and are satisfied by construction of $\hat{\pi}_{S'Q'T'}$.

Constraints (4.15c) are satisfied by a similar argument as above.

Constraints (4.15d) reduces to $\hat{\pi}_{S'Q'T'} + \hat{\pi}_{S'Q''T''} + \hat{\gamma} = \Delta^* + 1 - \Delta^* \geq 1$, and is clearly satisfied.

We next observe that $(\hat{\mu}, \hat{\eta}, \hat{\gamma}, \hat{\pi}) \geq 0$ by construction.

Dual objective value $\hat{d}$ is

$$\begin{aligned}
\hat{d} =& 2\sum_{SQT} |S|\hat{\pi}_{SQT} - \sum_{SQT:S_{odd}} \hat{\pi}_{SQT} + 1^T\hat{\mu} + 1^T\hat{\eta} + \hat{\gamma} \\
&= 2|S'|(\hat{\pi}_{S'Q'T'} + \hat{\pi}_{S'Q''T''}) + (c(C^+) - \sum_{S'} \Delta^*) + (d(D^+) - \sum_{S'} \Delta^*) + (1 - \Delta^*) \\
&= 2|S'|\Delta^* + c(C^+) + d(D^+) - 2\sum_{S} \Delta^* + 1 - \Delta^* \\
&= c(C^+) + d(D^+) + 1 - \Delta^*,
\end{aligned}$$

which is the same as $\hat{p}$.

(b) $c' + d' < \Delta^*$: We let $\rho = \Delta^* - c' - d'$, and $\alpha = \arg\min\{-c_k : k \in C^- \cap D^-\}$. Then

$c' = c_\alpha$. Let $\sigma_1, \sigma_2, \ldots, \sigma_p$ be an ordering of $C^- \cap D^- \setminus \{\alpha\}$ such that

$$d' = -d_{\sigma_1} \leq -d_{\sigma_2} \leq \cdots - d_{\sigma_p}.$$

We first observe that $\alpha \neq \sigma_1$, if not then $\Delta^* > c' + d' = -c_\alpha - d_\alpha$, but by definition of Δ^*, $\Delta^* \leq -c_\alpha - d_\alpha$. This implies that $C^- \cap D^- \setminus \{\alpha\}$ is not empty.

Let $l^* = \max_{j \in [p]} \{j \mid -d_{\sigma_j} + d_{\sigma_1} < \rho\}$.

We now do the following updates in $\hat{\pi}$:

$$\begin{aligned} &\hat{\pi}_{S'Q'T'} = c' \quad \text{where } S' = C^+ \cap D^+, Q' = C^-, T' = C^+ \cap D^- \\ &\hat{\pi}_{S'Q''T''} = d' \text{ where } S' = C^+ \cap D^+, Q'' = C^- \cap D^+, T'' = D^- \end{aligned} \tag{4.20}$$

With $S' = C^+ \cap D^+$, we let $Q_{j+1} = C^- \cap D^+ \cup \{\sigma_1, \sigma_2, \ldots, \sigma_j\}, T_{j+1} = D^- \setminus \{\sigma_1, \sigma_2, \ldots, \sigma_j\}$, we now update following dual variables $\hat{\pi}_1, \ldots, \hat{\pi}_{l^*+1}$.

$$\hat{\pi}_{j+1} = \hat{\pi}_{S'Q_{j+1}T_{j+1}} = -d_{\sigma_{j+1}} + d_{\sigma_j}, \quad \forall j = 0, l^* - 1 \tag{4.21a}$$

$$\hat{\pi}_{l^*+1} = \hat{\pi}_{S'Q_{l^*+1}T_{l^*+1}} = \rho - (-d_{\sigma_{l^*}} + d_{\sigma_1}) \tag{4.21b}$$

By construction in (4.21), $\sum_{j=1}^{l^*+1} \hat{\pi}_j = \rho$. This also implies $1^\top \hat{\pi} = \Delta^*$.

We now show that this dual assignment in (4.20),(4.21) is feasible with objective value $\hat{d} = \hat{p}$.

Constraints (4.15b):

- If $k \in C^+ \cap D^+$, then $k \in S'$, and constraints (4.15b) can be simplified to $\hat{\pi}_{S'Q'T'} + \hat{\pi}_{S'Q''T''} + \sum_{j=1}^{l^*+1} \hat{\pi}_{S'Q_jT_j} + \hat{\mu}_k = \Delta^* + (c_k - \Delta^*) \geq c_k$, and are clearly satisfied.
- If $k \in C^- \cap D^+$, then $k \in Q' \cap Q''$, and constraints (4.15b) can be simplified to $-\hat{\pi}_{S'Q'T'} - \hat{\pi}_{S'Q''T''} - \sum_{j=1}^{l^*+1} \hat{\pi}_{S'Q_jT_j} = -\Delta^* \geq c_k$. The later follows from the definition of Δ^*.
- If $k \in C^+ \cap D^-$, then $k \in T' \cap T''$, and constraints (4.15b) can be simplified to $\hat{\mu}_k \geq c_k$ and are satisfied since $\hat{\mu}_k = c_k \; \forall k \in C^+ \cap D^-$.
- If $k = \alpha$, then $k \in Q' \setminus Q''$, and left-hand-side of constraints (4.15b) can be simplified to $-\sum_{Q \ni k} \hat{\pi}_{SQT} = -c' \geq c_k$. The inequality follows by the definition of c'.
- If $k \in C^- \cap D^- \setminus \{\alpha\} = \{\sigma_1, \sigma_2, \ldots, \sigma_p\}$, then $k = \sigma_l$ for some $l \in \{1, 2, \ldots, p\}$, and

further observe that $k \in Q' \setminus Q''$. Constraints (4.15b) can be simplified to

$$\begin{aligned}
-\sum_{Q \ni k} \hat{\pi}_{SQT} &= -\hat{\pi}_{S'Q'T'} - \sum_{j=1:Q_j \ni k}^{l^*+1} \hat{\pi}_{S'Q_jT_j} \\
&= -c' - \sum_{j=l+1}^{p+1} \hat{\pi}_{S'Q_jT_j} \\
&= -c' - \rho + \sum_{j=1}^{l} \hat{\pi}_{S'Q_jT_j} \\
&\geq -c' - \rho + (-d_{\sigma_l} + d_{\sigma_1}) \\
&\geq c_{\sigma_l} = c_k
\end{aligned}$$

The last inequality follows because $-c_{\sigma_l} - d_{\sigma_l} \geq \Delta^* = c' + d' + \rho$ by definition of Δ^* and using $d' = -d_{\sigma_1}$.

Constraints (4.15c):

- If $k \in C^+ \cap D^+$, then constraints (4.15c) are valid by a similar argument as constraints (4.15b).
- If $k \in C^+ \cap D^-$, then $k \in T' \cap T''$ and constraints (4.15c) can be simplified to $-\hat{\pi}_{S'Q'T'} - \hat{\pi}_{S'Q''T''} - \sum_{j=1}^{l^*+1} \hat{\pi}_{S'Q_jT_j} = -\Delta^* \geq d_k$. The later follows by the definition of Δ^*.
- If $k \in C^- \cap D^+$, then $k \in Q' \cap Q''$, and constraints (4.15b) can be simplified to $\hat{\eta}_k \geq d_k$ and are satisfied since $\hat{\eta}_k = d_k \; \forall k \in C^- \cap D^+$.
- If $k = \alpha$, then $k \in T'' \setminus T'$ and $k \in T_j \; \forall j \in [l^* + 1]$, and left-hand-side of constraints (4.15c) can be simplified to $-\sum_{T \ni k} \hat{\pi}_{SQT} = \hat{\pi}_{S'Q''T''} - \sum_{j=1}^{l^*+1} \hat{\pi}_{S'Q_jT_j} = -d' - \rho \geq d_\alpha$. The later follows because $-c_\alpha - d_\alpha \geq \Delta^* = c' + d' + \rho$ and using $c' = -c_\alpha$.
- If $k \in C^- \cap D^- \setminus \{\alpha\} = \{\sigma_1, \sigma_2, \ldots, \sigma_p\}$, then $k = \sigma_l$ for some $l \in \{1, 2, \ldots, p\}$, and

$k \in T'' \setminus T'$. Left-hand-side of constraints (4.15c) an be simplified to

$$\begin{aligned}
-\sum_{T \ni k} \hat{\pi}_{SQT} &= -\hat{\pi}_{S'Q''T''} - \sum_{j=1:T_j \ni k}^{l^*+1} \hat{\pi}_{S'Q_jT_j} \\
&= -d' - \sum_{j=1}^{l} \hat{\pi}_{S'Q_jT_j} \\
&= -d' - (-d_{\sigma_l} + d_{\sigma_1}) \\
&= -d' - (-d_{\sigma_l} - d') \\
&= d_{\sigma_l} = d_k
\end{aligned}$$

Constraint (4.15d) reduce to $1^\top \hat{\pi} + \hat{\gamma} = \Delta^* + 1 - \Delta^* \geq 1$, and is clearly satisfied. Dual objective $\hat{d} = c(C^+) + d(D^+) + 1 - \Delta^*$ and follows a similar argument as for case (a).

We have thus shown that for any given objective, (P) admits a feasible integral solution $(\hat{u}, \hat{v}, \hat{z})$ having value $c(C^+) + d(D^+) + 1 - \Delta^*$ and (D) admits a feasible solution $(\hat{\pi}, \hat{\gamma}, \hat{\mu}, \hat{\eta})$ with a value $\hat{d} = \hat{p}$. □

4.3 Insights into SQT inequalities

In this section, we provide some insights in the novel SQT inequalities (4.12a), (4.12b) proposed in Section 4.2.3. In Section 4.3.1, we discuss how to derive SQT inequalities via lifting from a lower dimensional face of $\mathrm{conv}(\mathcal{I})$. We then discuss linear-time separation of SQT inequalities in Section 4.3.2.

4.3.1 Deriving SQT inequalities via lifting

In this section, we show that the inequalities (4.12a), (4.12b) proposed in Section 4.2.3, can be derived via lifting. In particular, for some $i \in [r]$, we let (S, Q, T) be a partition of $[r] \setminus i$, $\gamma = \bigoplus_{k \in [r] \setminus \{i\}} u_k v_k$, and derive facet-defining inequalities for the polytope $\mathrm{conv}(\mathcal{I} \cap \mathcal{F}_{SQT})$ where $\mathcal{F}_{SQT}$ is defined as follows:

$$\mathcal{F}_{SQT} = \{(u, v, z) \in \{0,1\}^{2r+1} : u_k = v_k = 1 \; \forall k \in S, u_k = 0 \; \forall k \in Q, v_k = 0, \; \forall k \in T\}. \tag{4.22}$$

We then lift the derived inequalities to obtain facet-defining inequalities for $\mathrm{conv}(\mathcal{I})$, which coincidentally are identical to (4.12a),(4.12b). The motivation for doing this is to

follow a similar strategy to derive valid inequalities for $\mathrm{conv}(\mathcal{I}_{ij} \cap \mathcal{I}_{i'j})$ for $i \neq i'$ as we discuss later in Section 4.4.1.

We rewrite the $\mathbb{F}_2$ vector dot product between u, v as follows:

$$z = u \otimes v = u_i v_i \oplus \gamma.$$

We now consider two different cases based on whether $|S|$ is even or odd.

- $|S|$ is even: In this case, $\gamma = 0$. The inequality $z \geq u_i + v_i - 1$ is facet defining for $\mathrm{conv}(\mathcal{I} \cap \mathcal{F}_{SQT})$. This is easy to observe since $\mathcal{I} \cap \mathcal{F}_{SQT}$ is the same as $\mathcal{I}$ with $r = 1$. Hence, by Proposition 4.5, for $r = 1$, dimension of polyhedron $\mathrm{conv}(\mathcal{F}_{SQT} \cap \mathcal{I})$ is 3, and dimension of face $F = \mathrm{conv}(\mathcal{F}_{SQT} \cap \mathcal{I}) \cap \{z = u_i + v_i - 1\}$ is 2 since $(1,0,0), (0,1,0), (1,1,1)$ are affinely independent points on F.

  We can now sequentially lift this inequality for variables fixed in S, Q, T to obtain a facet defining inequality for $\mathrm{conv}(\mathcal{I})$ (Prop. 7.2, [30]). In particular, we lift the following inequality.

$$u_i + v_i - z + \sum_{k \in S} \alpha_k (1 - u_k) + \sum_{k \in S} \beta_k (1 - v_k) + \sum_{Q} \alpha_k u_k + \sum_{T} \beta_k v_k \leq 1$$

  Here α, β are the lifting coefficients for u_k, v_k variables for $k \in [r] \setminus \{i\}$.

  - Let $S = \{1, 2, \ldots, |S|\}$. Lifting u_k in this order,

$$\begin{aligned}
\alpha_1 &= 1 - \max\{u_i + v_i - z \mid (u,v,z) \in \mathcal{I}, u_1 = 0, u_{S\setminus\{1\}} = v_S = 1, u_Q = v_T = 0\} \\
&= -1 \\
\alpha_2 &= 1 - \max\{u_i + v_i - z - (1 - u_1) \mid (u,v,z) \in \mathcal{I}, u_2 = 0, u_{S\setminus\{1,2\}} = v_S = 1, \\
&\qquad\qquad u_Q = v_T = 0\} \\
&= -1 \\
&\vdots \\
\alpha_{|S|} &= 1 - \max\{u_i + v_i - z - \sum_{k=1}^{|S|-1} (1 - u_k) \mid (u,v,z) \in \mathcal{I}, u_{|S|} = 0, v_S = 1, \\
&\qquad\qquad u_Q = v_T = 0\} \\
&= -1.
\end{aligned}$$

We now lift v_k for $k \in S$,

$$\begin{aligned}
\beta_1 &= 1 - \max\{u_i + v_i - z - \sum_{k\in S}(1-u_k) \mid (u,v,z) \in \mathcal{I}, v_1 = 0, v_{S\setminus\{1\}} = 1, \\
&\qquad\qquad u_Q = 0, v_T = 0\} \\
&= -1 \\
\beta_2 &= 1 - \max\{u_i + v_i - z - \sum_{k\in S}(1-u_k) - (1-v_1) \mid (u,v,z) \in \mathcal{I}, v_2 = 0, \\
&\qquad\qquad v_{S\setminus\{1,2\}} = 1, u_Q = 0, v_T = 0\} \\
&= -1 \\
&\vdots \\
\beta_{|S|} &= 1 - \max\{u_i + v_i - z - \sum_{k\in S}(1-u_k) - \sum_{k=1}^{|S|-1}(1-v_k) \mid (u,v,z) \in \mathcal{I}, v_{|S|} = 0, \\
&\qquad\qquad u_Q = v_T = 0\} \\
&= -1
\end{aligned}$$

– Let $Q = \{1, 2, \ldots, |Q|\}$. Lifting u_k in this order,

$$\begin{aligned}
\alpha_1 &= 1 - \max\{u_i + v_i - z - \sum_{k\in S}(2-u_k-v_k) \mid (u,v,z) \in \mathcal{I}, u_1 = 1, \\
&\qquad\qquad u_{Q\setminus\{1\}} = v_T = 0\} \\
&= -1 \\
\alpha_2 &= 1 - \max\{u_i + v_i - z - \sum_{k\in S}(2-u_k-v_k) - u_1 \mid (u,v,z) \in \mathcal{I}, u_2 = 1, \\
&\qquad\qquad u_{Q\setminus\{1,2\}} = v_T = 0\} \\
&= -1 \\
&\vdots \\
\alpha_{|Q|} &= 1 - \max\{u_i + v_i - z - \sum_{k\in S}(2-u_k-v_k) - \sum_{k=1}^{|Q|-1} u_k \mid (u,v,z) \in \mathcal{I}, u_{|S|} = 1, \\
&\qquad\qquad v_T = 0\} \\
&= -1
\end{aligned}$$

– Lifting in T follows a similar argument and $\beta_k = -1 \; \forall k \in T$.

The lifted inequality is then as follows:

$$u_i + v_i - z - \sum_{k \in S}(1 - u_k) - \sum_{k \in S}(1 - v_k) - \sum_{Q} u_k - \sum_{T} v_k \leq 1.$$

Let $S \leftarrow S \cup \{i\}$, with $|S|$ now becoming odd, the inequality can then be rewritten as

$$\begin{aligned} 2 - z - \sum_{k \in S}(1 - u_k) - \sum_{k \in S}(1 - v_k) - \sum_{Q} u_k - \sum_{T} v_k \leq 1, \\ - z + \sum_{k \in S}(u_k + v_k) - \sum_{Q} u_k - \sum_{T} v_k \leq 2|S| - 1, \end{aligned}$$

which is the same as (4.12b).

- $|S|$ is odd: In this case, $\gamma = 1$, and facet defining inequality for $\mathrm{conv}(\mathcal{I} \cap \mathcal{F}_{SQT})$ is $z \leq 2 - u_i - v_i$. Lifting this inequality follows an identical procedure to the previous case and gives inequalities of the form (4.12a).

Conjecture 4.13. (*i*) *Lifting coefficients for variables fixed in set* $S = \{k \mid u_k = v_k = 1\}$, $Q = \{k \mid u_k = 0\}$, *and* $T = \{k \mid v_k = 1\}$ *for inequality* $z \geq u_i + v_i - 1$ *are order independent.*

(*ii*) *Lifting coefficients for variables fixed in set* $S = \{k \mid u_k = v_k = 1\}$, $Q = \{k \mid u_k = 0\}$, *and* $T = \{k \mid v_k = 1\}$ *for inequality* $z \leq 2 - u_i - v_i$ *are order independent.*

4.3.2 Separating SQT inequalities

We now discuss a separation strategy for the SQT inequalities (4.12) proposed in Section 4.2.3. In particular, given a point $(\hat{u}, \hat{v}, \hat{z})$, we want to check if an inequality of type (4.12a) or (4.12b) is violated at $(\hat{u}, \hat{v}, \hat{z})$. We first formulate the separation problem as an integer program and then provide a closed-form for its solution.

We define decision variables $\alpha_k \in \{0, 1\}$ to be 1 if $k \in S$, $\beta_k \in \{0, 1\}$ to be 1 if $k \in Q$, and $\gamma_k \in \{0, 1\}$ to be 1 if $k \in T$, and 0 otherwise $\forall k \in \{1, 2, \ldots, r\}$. Note that $|S| = \sum_{k \in [r]} \alpha_k$

We can now formulate the separation problem for (4.12a) as follows:

$$\max \sum_{k=1}^{r} \Big[(\hat{u}_k + \hat{v}_k)\alpha_k - \hat{u}_k\beta_k - \hat{v}_k\gamma_k - 2\alpha_k\Big] \tag{4.23a}$$

$$\alpha_k + \beta_k + \gamma_k = 1 \qquad \forall k \in [r] \tag{4.23b}$$

$$\sum_{k\in[r]} \alpha_k = 2t \tag{4.23c}$$

$$\alpha, \beta, \gamma \in \{0,1\}^r, t \in \mathbb{Z}_+. \tag{4.23d}$$

Eliminating γ_k, we can alternatively write the above integer program as,

$$\max \sum_{k=1}^{r} \Big[(\hat{u}_k + 2\hat{v}_k - 2)\alpha_k + (\hat{v}_k - \hat{u}_k)\beta_k - \hat{v}_k\Big] \tag{4.24a}$$

$$\alpha_k + \beta_k \leq 1 \qquad \forall k \in [r] \tag{4.24b}$$

$$\sum_{k\in[r]} \alpha_k = 2t \tag{4.24c}$$

$$\alpha, \beta \in \{0,1\}^r, t \in \mathbb{Z}_+. \tag{4.24d}$$

A similar integer program can be constructed for separating (4.12b) by replacing constraint (4.23c) with $\sum_{k\in[r]} \alpha_k = 2t + 1$.

We next discuss a closed form solution for the above integer program. We let $c_1^k = \hat{u}_k + 2\hat{v}_k - 2$, $c_2^k = (\hat{v}_k - \hat{u}_k)$, and $c_3^k = -\hat{v}_k$ for $k \in [r]$. We then use Algorithm 5 to find a partition of $[r]$. In Algorithm 5, we first greedily assign each $k \in [r]$ so as to maximize the objective value of partition. If resulting $\hat{S}$ has even cardinality, we are done. If not, we modify set $\hat{S}$ by choosing the best option between expanding $\hat{S}$, i.e., adding an element to $\hat{S}$ or shrinking $\hat{S}$, i.e., removing an element from $\hat{S}$. For the generated partition from Algorithm 5, we then check if $c_1(\hat{S}) + c_2(\hat{Q}) + c_3(\hat{T}) > -\hat{z}$. If yes, we have found a violated inequality of type (4.12a). An identical procedure is used to separate odd SQT inequality (4.12b).

Theorem 4.14 shows that Algorithm 5 is exact.

Theorem 4.14. *The objective value of partition $(\hat{S}, \hat{Q}, \hat{T})$ constructed using Algorithm 5, i.e., $c_1(\hat{S}) + c_2(\hat{Q}) + c_3(\hat{T})$ is the same as the optimal value of integer program 4.24.*

Proof. Let (S^*, Q^*, T^*) be the solution to the integer program (4.24) with optimal value c^*. We let e denote the element that is swapped if $|\hat{S}|$ is odd at the end of line 10 in Algorithm 5.

We first claim that if $k \notin S^* \cap \hat{S}$, then either $k \in \hat{Q} \cap Q^*$ or $k \in \hat{T} \cap T^*$, or there exists another solution (S', Q', T') of value c^* such that $k \in Q' \cap \hat{Q}$ or $k \in T' \cap \hat{T}$. Assume $c_2^k \neq c_3^k$

Algorithm 5: Linear time separation for even SQT inequalities

Input: Vectors c_1, c_2, c_3

Output: Partition of $[r]$ in sets $\hat{S}, \hat{Q}, \hat{T}$ such that $|\hat{S}|$ is even and $(\hat{S}, \hat{Q}, \hat{T}) = \underset{(SQT)\in\mathcal{T}}{\arg\min}\, c_1(S) + c_2(Q) + c_3(T)$

$\hat{S} = \emptyset, \hat{Q} = \emptyset, \hat{T} = \emptyset$;
for $k \in [r]$ **do**
  if $c_1^k \geq c_2^k$ *and* $c_1^k \geq c_3^k$
    $\hat{S} \leftarrow \hat{S} \cup \{k\}$;
  else if $c_2^k \geq c_1^k$ *and* $c_2^k \geq c_3^k$
    $\hat{Q} \leftarrow \hat{Q} \cup \{k\}$;
  else
    $\hat{T} \leftarrow \hat{T} \cup \{k\}$;
  end
end
if $|\hat{S}|$ *not even*
  $c_{k_e} = \max\left\{ \max_{k\in Q}\{c_1^k - c_2^k\}, \max_{k\in T}\{c_1^k - c_3^k\} \right\}$; `/* Expand` $\hat{S}$ `*/`
  $c_{k_s} = \max\left\{ \max_{k\in S}\{-c_1^k + c_2^k\}, \max_{k\in S}\{-c_1^k + c_3^k\} \right\}$; `/* Shrink` $\hat{S}$ `*/`
  if $c_{k_e} \geq c_{k_s}$
    $\hat{S} \leftarrow \hat{S} \cup \{k_e\}$, $\hat{Q} \leftarrow \hat{Q} \setminus \{k_e\}$ if $-c_2^k \geq -c_3^k$ else $\hat{T} \leftarrow \hat{T} \setminus \{k_e\}$;
  else
    $\hat{S} \leftarrow \hat{S} \setminus \{k_s\}$, $\hat{Q} \leftarrow \hat{Q} \cup \{k_s\}$ if $c_2^k \geq c_3^k$ else $\hat{T} \leftarrow \hat{T} \cup \{k_s\}$;
  end
end
return $\hat{S}, \hat{Q}, \hat{T}$

and for contradiction, let $k \in \hat{Q}, k \notin Q^*$. Then, by construction $c_2^k \geq c_3^k \, \forall k \in \hat{Q}$, and if $k \notin Q^*$, one can improve the optimal value c^* by moving k from T^* to Q^*. Similarly, if $k \in Q^*, k \notin \hat{Q}$, then by construction of $\hat{Q}$ and $\hat{T}$, $c_3^k > c_2^k$. Thus, one can improve c^* by moving k from Q^* to T^*. In both cases, we assumed c^* to be the optimal value, a contradiction. If $c_2^k \neq c_3^k$, then c^* is unaffected by whether $k \in Q^*$ or $k \in T^*$, and hence one can construct Q' such that $k \in \hat{Q} \cap Q'$. A similar argument follows for sets $\hat{T}$ and T^*.

We next claim that if $k \in \hat{S}, k \notin S^*$, then either there exist another optimal solution (S', Q', T') of value c^* such that $k \in S' \cap \hat{S}$ or (S^*, Q^*, T^*) is not optimal.

If $k \in \hat{S} \setminus S^*$ and $k \neq e$, then $c_1^k \geq \max\{c_2^k, c_3^k\}$. Since $|\hat{S}|, |S|^*$ are even, at least one of the following is true: (i) $\exists k_1 \neq k, k_1 \in \hat{S}, k_1 \notin S^*$, (ii) $\exists k_2 \neq k, k_2 \in S^*, k_2 \notin S$. Case (i) results in two elements $k, k_1 \in \hat{S} \setminus S^*$ such that $c_1^k + c_1^{k_1} \geq \max\{c_2^k, c_3^k\} + \max\{c_2^{k_1}, c_3^{k_1}\}$ by the construction of $\hat{S}$. If the inequality holds, then it implies that c^* is not optimal since one

could improve c^* by moving k, k_1 to S^* from $Q^* \cup T^*$. If the equality holds, one can move k, k_1 to S^* without decreasing c^*, and construct a new solution (S', Q', T') with $k \in \hat{S} \cap S'$. In case (ii), since $k_2 \notin \hat{S}$, it follows that $c_1^{k_2} \leq \max\{c_2^{k_2}, c_3^{k_2}\}$ by construction. This implies that moving k from $Q^* \cup T^*$ to S^* and k_2 from S^* to $Q^* \cup T^*$ should either increase c^* or keep it the same. If c^* increases, then (S^*, Q^*, T^*) was not optimal, a contradiction. If c^* stays the same, we constructed a new solution (S', Q', T') of the value as c^* with $k \in \hat{S} \cap S'$.

If $k \in \hat{S} \setminus S^*$ and $k = e$, for $|S^*|$ to be even, at least one of the following is true:(i) $\exists k_1 \neq e, k_1 \in \hat{S}, k_1 \notin S^*$, (ii) $\exists k_2 \neq e, k_2 \in S^*, k_2 \notin S$. In case (i), since $e \in \hat{S}$, optimality of e in Algorithm 5 implies that $\max\{-c_1^{k_1} + c_2^{k_1}, -c_1^{k_1} + c_3^{k_1}\} \leq \max\{c_1^e - c_2^e, c_1^e - c_3^e\}$, or equivalently, $c_1^{k_1} + c_1^e \geq \max\{c_2^{k_1}, c_3^{k_1}\} + \max\{c_2^e, c_3^e\}$. Similar to $k \neq e$ argument, it then follows that either (S^*, Q^*, T^*) is not optimal or one can construct a new solution S', Q', T' of value c^* such that $e \in \hat{S} \cap S'$. In case (ii), since $k_2 \notin \hat{S}$, optimality of e in Algorithm 5 implies that $\max\{c_1^{k_2} - c_2^{k_2}, c_1^{k_2} - c_3^{k_2}\} \leq \max\{c_1^e - c_2^e, c_1^e - c_3^e\}$, or equivalently $c_1^e + \max\{c_2^{k_2}, c_3^{k_2}\} \geq \max\{c_2^e, c_3^e\} + c_1^{k_2}$. This implies that moving e from $Q^* \cup T^*$ to S^* and k_2 from S^* to $Q^* \cup T^*$ should either increase c^* or keep it the same. Similar to $k \neq e$ argument, it then follows that either (S^*, Q^*, T^*) is not optimal or one can construct a new solution S', Q', T' of value c^* such that $e \in \hat{S} \cap S'$.

We next claim that if $k \in S^*, k \notin \hat{S}$, then either there exist another optimal solution (S', Q', T') of value c^* such that $k \notin \hat{S} \cup S'$ or (S^*, Q^*, T^*) is not optimal. Argument for $k \in S^* \setminus \hat{S}$ is analogous to $k \in \hat{S} \setminus S^*$.

This completes the proof since we have shown that either (S^*, Q^*, T^*) is not optimal or one can construct a new solution (S', Q', T') of the same value as c^* such that $S' = \hat{S}, Q' = \hat{Q}, T' = \hat{T}$. □

4.4 Valid inequalities for the SQT formulation

In Section 4.2, we derived three formulations by formulating vector dot product for each row of U and V. In other words, we modeled the set $\mathcal{I}_{ij}$ $\forall i \in [d], j \in [n]$. In this Section, we choose $i, i' \in [d], i \neq i'$, and $j \in [n]$, and propose valid inequalities for $\text{conv}(\mathcal{I}_{ij} \cap \mathcal{I}_{i'j}) = \text{conv}(\{(u_i, u_{i'}, v_j, z_{ij}, z_{i'j}) \in \{0,1\}^{3r+2} \mid z_{ij} = u_i \otimes v_j, z_{i'j} = u_{i'} \otimes v_j\})$. We first present the inequalities in Section 4.4.1, discuss their derivations in Section 4.4.2, and discuss separation strategies in Section 4.4.3- 4.4.4.

4.4.1 Valid inequalities

For ease of notation, we refer to u_i as u, $u_{i'}$ as w, z_{ij} as z_u, and $z_{i'j}$ as z_w. We further let $\mathcal{I}_{uw} = \mathcal{I}_{ij} \cap \mathcal{I}_{i'j}$. Then,

$$\mathcal{I}_{uw} = \Big\{(u, w, v, z_u, z_w) \in \{0,1\}^{3r+2} \mid z_u = u \otimes v, z_w = w \otimes v\Big\}. \tag{4.25}$$

Let $[r'] = [r] \setminus \{i, j\}$ for some $i, j \in [r]$, $R_u = \bigoplus_{[r']} u_k v_k$, and $R_w = \bigoplus_{[r']} w_k v_k$. We next write system of equations corresponding to $\mathcal{I}_{uw}$ as follows:

$$z_u = u_i v_i \oplus u_j v_j \oplus R_u \tag{4.26a}$$
$$z_w = w_i v_i \oplus w_j v_j \oplus R_w. \tag{4.26b}$$

Theorem 4.15 (Family 1)**.** *Let (U, U^c), (W, W^c), and (V, V^c) be three bi-partitions of $[r'] = [r] \setminus \{i, j\}$ for some $i, j \in [r], i \neq j$ such that $U^c \cap W^c \cap V^c = \emptyset$.*

(*i*) *If $|U \cap V|$ is even and $|W \cap V|$ is even , then following is a valid inequality for* $\mathrm{conv}(\mathcal{I}_{uw})$*:*

$$(1 - w_j) + v_i + \sum_{U \cap V}(1 - u_k) + \sum_{U^c \cap V} u_k + \sum_{W \cap V}(1 - w_k) + \sum_{W^c \cap V} w_k + \sum_{(U \cup W) \cap V}(1 - v_k) + \sum_{(U \cup W) \cap V^c} v_k + (1 - z_u) + z_w \geq 1. \tag{4.27, F_1^{ee}}$$

(*ii*) *If $|U \cap V|$ is even and $|W \cap V|$ is odd , then following is a valid inequality for* $\mathrm{conv}(\mathcal{I}_{uw})$*:*

$$(1 - w_j) + v_i + \sum_{U \cap V}(1 - u_k) + \sum_{U^c \cap V} u_k + \sum_{W \cap V}(1 - w_k) + \sum_{W^c \cap V} w_k + \sum_{(U \cup W) \cap V}(1 - v_k) + \sum_{(U \cup W) \cap V^c} v_k + (1 - z_u) + (1 - z_w) \geq 1. \tag{4.28, F_1^{eo}}$$

(*iii*) *If $|U \cap V|$ is odd and $|W \cap V|$ is even , then following is a valid inequality for* $\mathrm{conv}(\mathcal{I}_{uw})$*:*

$$(1 - w_j) + v_i + \sum_{U \cap V}(1 - u_k) + \sum_{U^c \cap V} u_k + \sum_{W \cap V}(1 - w_k) + \sum_{W^c \cap V} w_k + \sum_{(U \cup W) \cap V}(1 - v_k) + \sum_{(U \cup W) \cap V^c} v_k + z_u + z_w \geq 1. \tag{4.29, F_1^{oe}}$$

(*iv*) *If* $|U \cap V|$ *is odd and* $|W \cap V|$ *is odd , then following is a valid inequality for* $\mathrm{conv}(\mathcal{I}_{uw})$*:*

$$(1-w_j)+v_i+\sum_{U\cap V}(1-u_k)+\sum_{U^c\cap V}u_k+\sum_{W\cap V}(1-w_k)+\sum_{W^c\cap V}w_k+ \sum_{(U\cup W)\cap V}(1-v_k)+\sum_{(U\cup W)\cap V^c}v_k+z_u+(1-z_w)\geq 1. \quad (4.30,\ F_1^{oo})$$

Proof. We show validity for part (i).Validity proofs for parts (ii)-(iv) are analogous. We prove the validity by showing that if a point violates (4.27) , then it is infeasible to (4.26). The inequality (4.27) is violated if and only if its left-hand-side $= 0$, i.e., $w_j = 1, v_i = 0, u_{U\cap V} = 1, u_{U^c\cap V} = 0, w_{W\cap V} = 1, w_{W^c\cap V} = 0, v_{(U\cup W)\cap V} = 1, v_{(U\cup W)\cap V^c} = 0, z_u = 1, z_w = 0$. We rewrite (4.26) by expanding R_u, R_w and substituting the above assignments as follows:

$$\overset{1}{\cancel{z_u}} = u_i\overset{0}{\cancel{v_i}} \oplus u_jv_j \bigoplus_{U\cap V} \overset{1}{\cancel{u_k}}\overset{1}{\cancel{v_k}} \bigoplus_{U^c\cap V} \overset{0}{\cancel{u_k}}v_k \bigoplus_{U\cap V^c} u_k\overset{0}{\cancel{v_k}} \bigoplus_{U^c\cap W\cap V^c} u_k\overset{0}{\cancel{v_k}} \bigoplus_{U^c\cap W^c\cap V^c} u_kv_k$$

$$\overset{0}{\cancel{z_w}} = w_i\overset{0}{\cancel{v_i}}\oplus\overset{1}{\cancel{w_j}}v_j \bigoplus_{W\cap V} \overset{1}{\cancel{w_k}}\overset{1}{\cancel{v_k}} \bigoplus_{W^c\cap V} \overset{0}{\cancel{w_k}}v_k \bigoplus_{W\cap V^c} w_k\overset{0}{\cancel{v_k}} \bigoplus_{U\cap W^c\cap V^c} w_k\overset{0}{\cancel{v_k}} \bigoplus_{U^c\cap W^c\cap V^c} w_kv_k.$$

Equivalently,

$$1 = u_jv_j \oplus (|U\cap V| \mod 2) \bigoplus_{U^c\cap W^c\cap V^c} u_kv_k$$
$$0 = \quad v_j \oplus (|W\cap V| \mod 2) \bigoplus_{U^c\cap W^c\cap V^c} w_kv_k.$$

Since $|U \cap V|$ and $|W \cap V|$ are both even, and $U^c \cap W^c \cap V^c = \emptyset$, the system reduces to

$$1 = u_j\ v_j$$
$$0 = \quad v_j.$$

Hence no feasible point to (4.26) violates the inequality (4.27). □

In the following theorem, we use the standard notation $U \triangle W = U \cup W \setminus (U \cap W)$ for the symmetric difference of sets.

Theorem 4.16 (Family 2)**.** *Let* (U, U^c)*,* (W, W^c)*, and* (V, V^c) *be three bi-partitions of* $[r'] = [r] \setminus \{i, j\}$ *for some* $i, j \in [r], i \neq j$ *such that* $U^c \cap W^c \cap V^c = \emptyset$ *and* $U \cap W \cap V^c = \emptyset$*.*

(*i*) *If $|U \cap V|$ is even and $|W \cap V|$ is even , then following is a valid inequality for* $\mathrm{conv}(\mathcal{I}_{uw})$*:*

$$(1-u_i)+(1-u_j)+(1-w_i)+(1-w_j)+\sum_{U\cap V}(1-u_k)+\sum_{U^c\cap V}u_k+\sum_{W\cap V}(1-w_k)+$$
$$\sum_{W^c\cap V}w_k+\sum_{(U\triangle W)\cap V}(1-v_k)+\sum_{(U\triangle W)\cap V^c}v_k+(1-z_u)+z_w\geq 1. \quad (4.31,\ F^2_{ee})$$

(*ii*) *If $|U \cap V|$ is even and $|W \cap V|$ is odd , then following is a valid inequality for* $\mathrm{conv}(\mathcal{I}_{uw})$*:*

$$(1-u_i)+(1-u_j)+(1-w_i)+(1-w_j)+\sum_{U\cap V}(1-u_k)+\sum_{U^c\cap V}u_k+\sum_{W\cap V}(1-w_k)+$$
$$\sum_{W^c\cap V}w_k+\sum_{(U\triangle W)\cap V}(1-v_k)+\sum_{(U\triangle W)\cap V^c}v_k+(1-z_u)+(1-z_w)\geq 1. \quad (4.32,\ F^2_{eo})$$

(*iii*) *If $|U \cap V|$ is odd and $|W \cap V|$ is even , then following is a valid inequality for* $\mathrm{conv}(\mathcal{I}_{uw})$*:*

$$(1-u_i)+(1-u_j)+(1-w_i)+(1-w_j)+\sum_{U\cap V}(1-u_k)+\sum_{U^c\cap V}u_k+\sum_{W\cap V}(1-w_k)+$$
$$\sum_{W^c\cap V}w_k+\sum_{(U\triangle W)\cap V}(1-v_k)+\sum_{(U\triangle W)\cap V^c}v_k+z_u+z_w\geq 1. \quad (4.33,\ F^2_{oe})$$

(*iv*) *If $|U \cap V|$ is odd and $|W \cap V|$ is odd , then following is a valid inequality for* $\mathrm{conv}(\mathcal{I}_{uw})$*:*

$$(1-u_i)+(1-u_j)+(1-w_i)+(1-w_j)+\sum_{U\cap V}(1-u_k)+\sum_{U^c\cap V}u_k+\sum_{W\cap V}(1-w_k)+$$
$$\sum_{W^c\cap V}w_k+\sum_{(U\triangle W)\cap V}(1-v_k)+\sum_{(U\triangle W)\cap V^c}v_k+z_u+(1-z_w)\geq 1. \quad (4.34,\ F^2_{oo})$$

Proof. We show validity for part (i). Validity proofs for parts (ii)-(iv) are analogous. We prove validity by showing that if a point violates (4.31), then it is infeasible to (4.26). The inequality (4.31) is violated if and only if its left-hand-side $= 0$, i.e., $u_i = 1, u_j = 1, w_i = 1, w_j = 1, u_{U\cap V} = 1, u_{U^c\cap V} = 0, w_{W\cap V} = 1, w_{W^c\cap V} = 0, v_{(U\triangle W)\cap V} = 1, v_{(U\triangle W)\cap V^c} = 0, z_u = 1, z_w = 0$.

We rewrite (4.26) by expanding R_u, R_w and substituting the above assignments as follows:

$$\overset{1}{\cancel{z_u}} = \overset{1}{\cancel{w_i}}v_i \oplus \overset{1}{\cancel{w_j}}v_j \bigoplus_{U\cap W\cap V} \overset{1}{\cancel{w_k}}v_k \bigoplus_{U\cap W^c\cap V} \overset{1}{\cancel{w_k}}\overset{1}{\cancel{v_k}} \bigoplus_{U^c\cap V} \overset{0}{\cancel{w_k}}v_k \bigoplus_{U\cap W\cap V^c} u_kv_k \bigoplus_{U\cap W^c\cap V^c} u_k\overset{0}{\cancel{v_k}} \bigoplus_{U^c\cap W\cap V^c} u_k\overset{0}{\cancel{v_k}} \bigoplus_{U^c\cap W^c\cap V^c} u_kv_k$$

$$\overset{0}{\cancel{z_w}} = \overset{1}{\cancel{w_i}}v_i \oplus \overset{1}{\cancel{w_j}}v_j \bigoplus_{U\cap W\cap V} \overset{1}{\cancel{w_k}}v_k \bigoplus_{U^c\cap W\cap V} \overset{1}{\cancel{w_k}}\overset{1}{\cancel{v_k}} \bigoplus_{W^c\cap V} \overset{0}{\cancel{w_k}}v_k \bigoplus_{U\cap W\cap V^c} w_kv_k \bigoplus_{U^c\cap W\cap V^c} w_k\overset{0}{\cancel{v_k}} \bigoplus_{U\cap W^c\cap V^c} w_k\overset{0}{\cancel{v_k}} \bigoplus_{U^c\cap W^c\cap V^c} w_kv_k.$$

Equivalently,

$$1 = v_i \oplus v_j \bigoplus_{U\cap W\cap V} v_k \oplus (|U \cap W^c \cap V| \mod 2) \bigoplus_{U\cap W\cap V^c} u_k v_k \bigoplus_{U^c\cap W^c\cap V^c} u_k v_k$$
$$0 = v_i \oplus v_j \bigoplus_{U\cap W\cap V} v_k \oplus (|U^c \cap W \cap V| \mod 2) \bigoplus_{U\cap W\cap V^c} u_k v_k \bigoplus_{U^c\cap W^c\cap V^c} u_k v_k.$$

We first point out that $|U \cap W^c \cap V| \mod 2 = |U^c \cap W \cap V| \mod 2 = |U \cap W \cap V| \mod 2$. This is because $U \cap W^c \cap V = (U \setminus (U \cap W)) \cap V = (U \cap V) \setminus (U \cap W \cap V)$. Since $U \cap V$ is even, $|U \cap V| \mod 2 = 0$, and $|U \cap W^c \cap V| \mod 2 = |U \cap W \cap V| \mod 2$. Set $U^c \cap W \cap V$ follows an identical argument. The above system of equations can then be equivalently written as:

$$1 = v_i \oplus v_j \bigoplus_{U\cap W\cap V} v_k \oplus (|U \cap W \cap V| \mod 2) \bigoplus_{U\cap W\cap V^c} u_k v_k \bigoplus_{U^c\cap W^c\cap V^c} u_k v_k$$
$$0 = v_i \oplus v_j \bigoplus_{U\cap W\cap V} v_k \oplus (|U \cap W \cap V| \mod 2) \bigoplus_{U\cap W\cap V^c} u_k v_k \bigoplus_{U^c\cap W^c\cap V^c} u_k v_k.$$

Since $U \cap W \cap V^c$ and $U^c \cap W^c \cap V^c$ are empty sets, the above system of equations is infeasible. Hence, no feasible point to (4.26) violates inequality (4.31). □

Theorem 4.17 (Family 3)**.** *Let (U, U^c), (W, W^c), and (V, V^c) be three bi-partitions of $[r'] = [r] \setminus \{i, j\}$ for some $i, j \in [r], i \neq j$ such that $U^c \cap W^c \cap V^c = \emptyset$ and $U \cap V^c = \emptyset$.*

(*i*) *If $|U \cap V|$ is even and $|W \cap V|$ is even , then following is a valid inequality for* $\mathrm{conv}(\mathcal{I}_{uw})$*:*

$$(1-u_i)+(1-u_j)+v_i+v_j+\sum_{U\cap V}(1-u_k)+\sum_{U^c\cap V}u_k+2\sum_{W\cap V}(1-w_k)+2\sum_{W^c\cap V}w_k+$$
$$\sum_{U\cap V}(1-v_k)+2\sum_{U^c\cap W\cap V}(1-v_k)+2\sum_{U^c\cap W\cap V^c}v_k+z_u+2(1-z_w)\geq 2. \quad (4.35,\ F_3^{ee})$$

(*ii*) *If $|U \cap V|$ is even and $|W \cap V|$ is odd, then following is a valid inequality for* $\mathrm{conv}(\mathcal{I}_{uw})$*:*

$$(1-u_i)+(1-u_j)+v_i+v_j+\sum_{U\cap V}(1-u_k)+\sum_{U^c\cap V}u_k+2\sum_{W\cap V}(1-w_k)+2\sum_{W^c\cap V}w_k+$$
$$\sum_{U\cap V}(1-v_k)+2\sum_{U^c\cap W\cap V}(1-v_k)+2\sum_{U^c\cap W\cap V^c}v_k+z_u+2z_w\geq 2. \quad (4.36,\ F_3^{eo})$$

(*iii*) *If $|U \cap V|$ is odd and $|W \cap V|$ is even, then following is a valid inequality for* $\mathrm{conv}(\mathcal{I}_{uw})$*:*

$$(1-u_i)+(1-u_j)+v_i+v_j+\sum_{U\cap V}(1-u_k)+\sum_{U^c\cap V}u_k+2\sum_{W\cap V}(1-w_k)+2\sum_{W^c\cap V}w_k+$$
$$\sum_{U\cap V}(1-v_k)+2\sum_{U^c\cap W\cap V}(1-v_k)+2\sum_{U^c\cap W\cap V^c}v_k+(1-z_u)+2(1-z_w)\geq 2. \tag{4.37, F_3^{oe}}$$

(*iv*) *If $|U \cap V|$ is odd and $|W \cap V|$ is odd, then following is a valid inequality for* $\mathrm{conv}(\mathcal{I}_{uw})$*:*

$$(1-u_i)+(1-u_j)+v_i+v_j+\sum_{U\cap V}(1-u_k)+\sum_{U^c\cap V}u_k+2\sum_{W\cap V}(1-w_k)+2\sum_{W^c\cap V}w_k+$$
$$\sum_{U\cap V}(1-v_k)+2\sum_{U^c\cap W\cap V}(1-v_k)+2\sum_{U^c\cap W\cap V^c}v_k+(1-z_u)+2z_w\geq 2. \tag{4.38, F_3^{oo}}$$

Proof. We show validity for part (i). Validity proofs for parts (ii)-(iv) are analogous. We prove validity by showing that if a point violates inequality (4.35), then it is infeasible to (4.26). The above inequality (4.35) is violated if and only if its left-hand-side $\in \{0, 1\}$.

- Left-hand-side$= 0$: We need to fix all variable with negative coefficients to 1 and with positive coefficients to 0, i.e., $u_i = 1, u_j = 1, v_i = 0, v_j = 0, u_{U\cap V} = 1, u_{U^c\cap V} = 0, w_{W\cap V} = 1, w_{W^c\cap V} = 0, v_{U\cap V} = 1, v_{U\cap V^c} = 0, v_{U^c\cap W\cap V} = 1, v_{U^c\cap W\cap V^c} = 0, z_u = 0, z_w = 1$. We rewrite (4.26) by expanding R_u, R_w, and substituting the above assignments as follows:

$$\overset{0}{\cancel{z_u}} = \overset{1}{\cancel{u_i}}\overset{0}{\cancel{v_i}} \oplus \overset{1}{\cancel{u_j}}\overset{0}{\cancel{v_j}} \bigoplus_{U\cap V} \overset{1}{\cancel{u_k}}\overset{1}{\cancel{v_k}} \bigoplus_{U^c\cap V} \overset{0}{\cancel{u_k}}v_k \bigoplus_{U\cap V^c} u_kv_k \bigoplus_{U^c\cap W\cap V^c} u_k\overset{0}{\cancel{v_k}} \bigoplus_{U^c\cap W^c\cap V^c} u_kv_k$$
$$\overset{1}{\cancel{z_w}} = w_i\overset{0}{\cancel{v_i}} \oplus w_j\overset{0}{\cancel{v_j}} \bigoplus_{W\cap V} \overset{1}{\cancel{w_k}}\overset{1}{\cancel{v_k}} \bigoplus_{W^c\cap V} \overset{0}{\cancel{w_k}}v_k \bigoplus_{U\cap V^c} w_kv_k \bigoplus_{U^c\cap W\cap V^c} w_k\overset{0}{\cancel{v_k}} \bigoplus_{U^c\cap W^c\cap V^c} w_kv_k.$$

  Equivalently,

$$0 = (|U\cap V| \mod 2) \bigoplus_{U\cap V^c} u_kv_k \bigoplus_{U^c\cap W^c\cap V^c} u_kv_k$$
$$1 = (|W\cap V| \mod 2) \bigoplus_{U\cap V^c} w_kv_k \bigoplus_{U^c\cap W^c\cap V^c} w_kv_k.$$

  The above is infeasible since both $|U \cap V|$ and $|W \cap V|$ are even, and $U \cap V^c$ and $U^c \cap W^c \cap V^c$ are empty by construction.

- Left-hand-side$= 1$: For left-hand-side to be 1, all variables with a coefficient of $+2$ and -2 need to be 0, i.e., $w_{W\cap V} = 1, w_{W^c\cap V} = 0, v_{U^c\cap W\cap V} = 1, v_{U^c\cap W\cap V^c} = 0, z_w = 1$.

The inequality then reduces to

$$(1-u_i)+(1-u_j)+v_i+v_j+\sum_{U\cap V}(1-u_k)+\sum_{U^c\cap V}u_k+\sum_{U\cap V}(1-v_k)+z_u\geq 2, \quad (4.39)$$

and the system of equations (4.26) reduces to

$$z_u = u_iv_i \oplus u_jv_j \bigoplus_{U\cap V} u_kv_k \bigoplus_{U^c\cap W\cap V} u_k\overset{1}{\cancel{v_k}} \bigoplus_{U^c\cap W^c\cap V} u_kv_k \bigoplus_{U\cap V^c} u_kv_k \bigoplus_{U^c\cap W\cap V^c} u_k\overset{0}{\cancel{v_k}} \bigoplus_{U^c\cap W^c\cap V^c} u_kv_k$$
$$\overset{1}{\cancel{z_w}} = w_iv_i\oplus w_jv_j \bigoplus_{U\cap W\cap V} \overset{1}{\cancel{w_k}}v_k \bigoplus_{U^c\cap W\cap V} \overset{1}{\cancel{w_k}}\overset{1}{\cancel{v_k}} \bigoplus_{W^c\cap V} \overset{0}{\cancel{w_k}}v_k \bigoplus_{U\cap V^c} w_kv_k \bigoplus_{U^c\cap W\cap V^c} w_k\overset{0}{\cancel{v_k}} \bigoplus_{U^c\cap W^c\cap V^c} w_kv_k.$$

Equivalently,

$$z_u = u_iv_i \oplus u_jv_j \bigoplus_{U\cap V} u_kv_k \bigoplus_{U^c\cap W\cap V} u_k \bigoplus_{U^c\cap W^c\cap V} u_kv_k \bigoplus_{U\cap V^c} u_kv_k \bigoplus_{U^c\cap W^c\cap V^c} u_kv_k \quad (4.40a)$$
$$1 = w_iv_i\oplus w_jv_j \oplus (|U^c\cap W\cap V| \mod 2) \bigoplus_{U\cap W\cap V} v_k \bigoplus_{U\cap V^c} w_kv_k \bigoplus_{U^c\cap W^c\cap V^c} w_kv_k. \quad (4.40b)$$

Left-hand-side is 1 if all but one variable with a coefficient of $+1$ takes a value of 0 or if all but one variable with a coefficient of -1 takes a value of 1. We first make left-hand-side to be 0 by fixing all the variables with a coefficient of $+1$ to 0 and variables with a coefficient of -1 to 1. We then flip one variable at a time to make left-hand-side 1 and analyze the resulting system for feasibility. Fixing $u_i=1, u_j=1, v_i=0, v_j=0, u_{U\cap V}=1, u_{U^c\cap V}=0, v_{U\cap V}=1, z_u=0$, and using the fact that $U\cap V^c$ and $U^c\cap W^c\cap V^c$ are $\emptyset$, above system of equations (4.40) reduces to

$$\overset{0}{\cancel{z_u}} = \overset{1}{\cancel{u_i}}\overset{0}{\cancel{v_i}} \oplus \overset{1}{\cancel{u_j}}\overset{0}{\cancel{v_j}} \bigoplus_{U\cap V} \overset{1}{\cancel{u_k}}\overset{1}{\cancel{v_k}} \bigoplus_{U^c\cap W\cap V} \overset{0}{\cancel{u_k}} \bigoplus_{U^c\cap W^c\cap V} \overset{0}{\cancel{u_k}}v_k \quad (4.41a)$$
$$1 = w_i\overset{0}{\cancel{v_i}} \oplus w_j\overset{0}{\cancel{v_j}} \oplus (|U^c\cap W\cap V| \mod 2) \bigoplus_{U\cap W\cap V} \overset{1}{\cancel{v_k}}. \quad (4.41b)$$

Currently, the above system is identical to the system in previous case (left-hand-side$=0$). We now try to change the value of exactly one variable to make left-hand-side as 1 for (4.39). Observe that the first equation (4.41a) is currently feasible ($|U\cap V|$ is even) while second (4.41b) is not since $|(U^c\cap W\cap V)\cup(U\cap W\cap V)| = |W\cap V|$ is even resulting in RHS for (4.41b) to be 0. Clearly, only flipping one variable in $\{u_i, u_j, u_{U\cap V}, u_{U^c\cap V}\}$ will not fix infeasibility since that change is restricted to (4.41a) and has no effect on (4.41b). The remaining variables are v_k variables for $k\in\{i,j,U\cap W\cap V\}$. Changing value of either of these variables will fix the infeasibility for (4.41b) but will make (4.41a) infeasible, thus leaving the system still

infeasible.

We have thus shown that no feasible point to system of equations (4.26) violates inequality (4.35). □

Theorem 4.18 (Family 4)**.** *Let* (U, U^c)*,* (W, W^c)*, and* (V, V^c) *be three bi-partitions of* $[r'] = [r] \setminus \{i, j\}$ *for some* $i, j \in [r], i \neq j$ *such that* $U^c \cap W^c \cap V^c = \emptyset$ *and* $U \cap V^c = \emptyset$*.*

(*i*) *If* $|U \cap V|$ *is even and*$|W \cap V|$ *is even , then following is a valid inequality for* $\operatorname{conv}(\mathcal{I}_{uw})$*:*

$$(1-u_i)+(1-u_j)+2(1-w_i)+(1-v_i)+v_j+\sum_{U\cap V}(1-u_k)+\sum_{U^c\cap V}u_k+2\sum_{W\cap V}(1-w_k)+$$
$$2\sum_{W^c\cap V}w_k+\sum_{U\cap V}(1-v_k)+2\sum_{U^c\cap W\cap V}(1-v_k)+2\sum_{U^c\cap W\cap V^c}v_k+(1-z_u)+2z_w\geq 2. \tag{4.42, F_4^{ee}}$$

(*ii*) *If* $|U \cap V|$ *is even and* $|W \cap V|$ *is odd, then following is a valid inequality for* $\operatorname{conv}(\mathcal{I}_{uw})$*:*

$$(1-u_i)+(1-u_j)+2(1-w_i)+(1-v_i)+v_j+\sum_{U\cap V}(1-u_k)+\sum_{U^c\cap V}u_k+2\sum_{W\cap V}(1-w_k)+$$
$$2\sum_{W^c\cap V}w_k+\sum_{U\cap V}(1-v_k)+2\sum_{U^c\cap W\cap V}(1-v_k)+2\sum_{U^c\cap W\cap V^c}v_k+(1-z_u)+2(1-z_w)\geq 2. \tag{4.43, F_4^{eo}}$$

(*iii*) *If* $|U \cap V|$ *is odd and* $|W \cap V|$ *is even, then following is a valid inequality for* $\operatorname{conv}(\mathcal{I}_{uw})$*:*

$$(1-u_i)+(1-u_j)+2(1-w_i)+(1-v_i)+v_j+\sum_{U\cap V}(1-u_k)+\sum_{U^c\cap V}u_k+2\sum_{W\cap V}(1-w_k)+$$
$$2\sum_{W^c\cap V}w_k+\sum_{U\cap V}(1-v_k)+2\sum_{U^c\cap W\cap V}(1-v_k)+2\sum_{U^c\cap W\cap V^c}v_k+z_u+2z_w\geq 2. \tag{4.44, F_4^{oe}}$$

(*iv*) *If* $|U \cap V|$ *is odd and* $|W \cap V|$ *is odd, then following is a valid inequality for* $\operatorname{conv}(\mathcal{I}_{uw})$*:*

$$(1-u_i)+(1-u_j)+2(1-w_i)+(1-v_i)+v_j+\sum_{U\cap V}(1-u_k)+\sum_{U^c\cap V}u_k+2\sum_{W\cap V}(1-w_k)+$$
$$2\sum_{W^c\cap V}w_k+\sum_{U\cap V}(1-v_k)+2\sum_{U^c\cap W\cap V}(1-v_k)+2\sum_{U^c\cap W\cap V^c}v_k+z_u+2(1-z_w)\geq 2. \tag{4.45, F_4^{oo}}$$

Proof. We show validity for part (i). Validity proofs for parts (ii)-(iv) are analogous. We prove validity by showing that if a point violates (4.42), then it is infeasible to (4.26). The above inequality (4.42) is violated if and only if left-hand-side$\in \{0, 1\}$.

- Left-hand-side$= 0$: Left-hand-side is 0 if we fix all variable with negative coefficients to 1 and with positive coefficients to 0 in (4.42), i.e, $u_i = 1, u_j = 1, w_i = 1, v_i = 1, v_j = 0, u_{U\cap V} = 1, u_{U^c\cap V} = 0, w_{W\cap V} = 1, w_{W^c\cap V} = 0, v_{U\cap V} = 1, v_{U^c\cap W\cap V} = 1, v_{U^c\cap W\cap V^c} = 0, z_u = 1, z_w = 0$.

  We rewrite (4.26) by expanding R_u, R_w, and substituting the above assignments as follows:

$$\overset{1}{\cancel{z_u}} = \overset{1}{\cancel{u_i}}\overset{1}{\cancel{v_i}} \oplus \overset{1}{\cancel{u_j}}\overset{0}{\cancel{v_j}} \bigoplus_{U\cap V} \overset{1}{\cancel{u_k}}\overset{1}{\cancel{v_k}} \bigoplus_{U^c\cap V} \overset{0}{\cancel{u_k}} v_k \bigoplus_{U\cap V^c} u_k v_k \bigoplus_{U^c\cap W\cap V^c} u_k \overset{0}{\cancel{v_k}} \bigoplus_{U^c\cap W^c\cap V^c} u_k v_k$$

$$\overset{0}{\cancel{z_w}} = \overset{1}{\cancel{w_i}}\overset{1}{\cancel{v_i}} \oplus w_j \overset{0}{\cancel{v_j}} \bigoplus_{W\cap V} \overset{1}{\cancel{w_k}}\overset{1}{\cancel{v_k}} \bigoplus_{W^c\cap V} \overset{0}{\cancel{w_k}} v_k \bigoplus_{U\cap V^c} w_k v_k \bigoplus_{U^c\cap W\cap V^c} w_k \overset{0}{\cancel{v_k}} \bigoplus_{U^c\cap W^c\cap V^c} w_k v_k.$$

  Equivalently,

$$1 = 1 \oplus (|U \cap V| \mod 2) \bigoplus_{U\cap V^c} u_k v_k \bigoplus_{U^c\cap W^c\cap V^c} u_k v_k$$

$$0 = 1 \oplus (|W \cap V| \mod 2) \bigoplus_{U\cap V^c} w_k v_k \bigoplus_{U^c\cap W^c\cap V^c} w_k v_k.$$

  The above system of equations is infeasible since both $|U \cap V|$ and $|W \cap V|$ are even, and $U \cap V^c$ and $U^c \cap W^c \cap V^c$ are empty by construction.

- Left-hand-side$= 1$: For left-hand-side to be 1, all variables with a coefficient of $+2$ and -2 need to be 0, i.e., $w_{W\cap V} = 1, w_{W^c\cap V} = 0, v_{U^c\cap W\cap V} = 1, v_{U^c\cap W\cap V^c} = 0, z_u = 0, z_w = 1$. The inequality (4.42) then reduces to

$$(1-u_i)+(1-u_j)+(1-v_i)+v_j+\sum_{U\cap V}(1-u_k)+\sum_{U^c\cap V} u_k+\sum_{U\cap V}(1-v_k)+(1-z_u) \geq 2, \quad (4.46)$$

  and the system of equations (4.26) reduces to

$$z_u = u_i v_i \oplus u_j v_j \bigoplus_{U\cap V} u_k v_k \bigoplus_{U^c\cap W\cap V} u_k \overset{1}{\cancel{v_k}} \bigoplus_{U^c\cap W^c\cap V} u_k v_k \bigoplus_{U\cap V^c} u_k v_k \bigoplus_{U^c\cap W\cap V^c} u_k \overset{0}{\cancel{v_k}} \bigoplus_{U^c\cap W^c\cap V^c} u_k v_k$$

$$\overset{0}{\cancel{z_w}} = \overset{1}{\cancel{w_i}} v_i \oplus w_j v_j \bigoplus_{U^c\cap W\cap V} \overset{1}{\cancel{w_k}}\overset{1}{\cancel{v_k}} \bigoplus_{U\cap W\cap V} \overset{1}{\cancel{w_k}} v_k \bigoplus_{W^c\cap V} \overset{0}{\cancel{w_k}} v_k \bigoplus_{U^c\cap W\cap V^c} w_k \overset{0}{\cancel{v_k}} \bigoplus_{U\cap V^c} w_k v_k \bigoplus_{U^c\cap W^c\cap V^c} w_k v_k.$$

  Using the fact that $U \cap V^c$ and $U^c \cap W^c \cap V^c$ are empty, we equivalently write the

above system as

$$z_u = u_i v_i \oplus u_j v_j \underset{U\cap V}{\oplus} u_k v_k \underset{U^c\cap W\cap V}{\oplus} u_k \underset{U^c\cap W^c\cap V}{\oplus} u_k v_k \tag{4.47a}$$

$$0 = \quad v_i \oplus w_j v_j \oplus (|U^c \cap W \cap V| \mod 2) \underset{U\cap W\cap V}{\oplus} v_k. \tag{4.47b}$$

Left-hand-side of inequality (4.46) is 1 if all but one variable with a coefficient of $+1$ takes a value of 0 or if all but one variable with a coefficient of -1 takes a value of 1. We first make left-hand-side 0 by fixing all the variables with a coefficient of $+1$ to 0 and variables with a coefficient of -1 to 1. We then change one variable at a time to make left-hand-side 1 and analyze the system for feasibility. Fixing $u_i = 1, u_j = 1, v_i = 1, v_j = 0, u_{U\cap V} = 1, u_{U^c\cap V} = 0, v_{U\cap V} = 1, z_u = 1$, above system of equations (4.47) reduces to

$$\cancel{\overset{1}{z_u}} = \cancel{\overset{1}{u_i}\overset{1}{v_i}} \oplus \cancel{\overset{1}{u_j}\overset{0}{v_j}} \underset{U\cap V}{\oplus} \cancel{\overset{1}{u_k}\overset{1}{v_k}} \underset{U^c\cap V}{\oplus} \cancel{\overset{0}{u_k}} v_k \tag{4.48a}$$

$$0 = \cancel{\overset{1}{v_i}} \oplus w_j \cancel{\overset{0}{v_j}} \oplus (|U^c \cap W \cap V| \mod 2) \underset{U\cap W\cap V}{\oplus} \cancel{\overset{1}{v_k}}. \tag{4.48b}$$

Currently, the above system (4.48) is identical to the system in previous case (left-hand-side$= 0$). We now change the value for exactly one variable to make left-hand-side as 1 for (4.46). Observe that the first equation (4.48a) is currently feasible ($|U\cap V|$ is even) while second (4.48b) is not since $|(U^c \cap W \cap V) \cup (U \cap W \cap V)| = |W \cap V|$ is even resulting in RHS for (4.48b) to be 1. Clearly, only changing value of one variable in $\{u_i, u_j, u_{U\cap V}, u_{U^c\cap V}\}$ will not fix infeasibility since that change is restricted to (4.48a) and has no effect on (4.48b). The remaining variables are v_k variables for $k \in \{i, j, U \cap W \cap V\}$. Flipping either of these will fix the infeasibility for (4.48b) but will make (4.48a) infeasible, thus leaving the system still infeasible.

We have thus shown that no feasible point to system of equations (4.26) violates inequality (4.42). □

4.4.2 Derivation of valid inequalities

While the validity of the inequalities proposed in Section 4.4.1 is established in that section, the proofs do not give insight into how the inequalities were derived. Thus, in this section we describe how we derived these families of inequalities. We use the same notation as in Section 4.4.1 and refer to rows of the matrix U as $u \in \{0,1\}^r$ and $w \in \{0,1\}^r$, and the corresponding z variables as z_u and z_w.

We next make a note on dim$(\text{conv}(\mathcal{I}_{uw}))$.

Proposition 4.19. *dim*$(\text{conv}(\mathcal{I}_{uw}))$=$3r+2$

We construct $3r+3$ affinely independent points (u, w, v, z_u, z_w) in $\mathcal{I}_u \cap \mathcal{I}_w$:

$$\begin{pmatrix} u \\ w \\ v \\ z_u \\ z_w \end{pmatrix} = \left[\begin{array}{c|ccc|c|ccc|c|cc} \mathbf{0}^\top & e_1^\top & e_2^\top \dots & e_r^\top & \mathbf{0}^\top & \mathbf{0}^\top \dots \mathbf{0}^\top & \mathbf{0}^\top & \mathbf{0}^\top \dots \mathbf{0}^\top & e_1^\top & \mathbf{0}^\top \\ \mathbf{0}^\top & \mathbf{0}^\top & \mathbf{0}^\top \dots & \mathbf{0}^\top & e_1^\top & e_2^\top \dots e_r^\top & \mathbf{0}^\top & \mathbf{0}^\top \dots \mathbf{0}^\top & \mathbf{0}^\top & e_1^\top \\ \mathbf{0}^\top & \mathbf{0}^\top & \mathbf{0}^\top \dots & \mathbf{0}^\top & \mathbf{0}^\top & \mathbf{0}^\top \dots \mathbf{0}^\top & e_1^\top & e_2^\top \dots e_r^\top & e_1^\top & e_1^\top \\ 0 & 0 & 0 \dots & 0 & 0 & 0 \dots 0 & 0 & 0 \dots 0 & 1 & 0 \\ 0 & 0 & 0 \dots & 0 & 0 & 0 \dots 0 & 0 & 0 \dots 0 & 0 & 1 \end{array}\right]$$

Each column of the above matrix represents an affinely independent point. This implies that dimension of $\text{conv}(\mathcal{I}_{uw})$ is $3r+3-1$. □

Let (U, U^c),(W, W^c), and (V, V^c) be three bi-partitions of $[r']$, i.e., $U \cap U^c = \emptyset, U \cup U^c = [r] \setminus \{i, j\}$, and similarly for the other two pairs (W, W^c) and (V, V^c). We then fix the following variables:

$$u_k = 1 \; \forall k \in U, \qquad u_k = 0 \; \forall k \in U^c \tag{4.49a}$$

$$w_k = 1 \; \forall k \in W, \qquad w_k = 0 \; \forall k \in W^c, \tag{4.49b}$$

$$v_k = 1 \; \forall k \in V, \qquad v_k = 1 \; \forall k \in V^c \tag{4.49c}$$

Let $|U \cap V|$ and $|W \cap V|$ be even resulting in $R_u = 0$ and $R_w = 0$. Note that R_u, and similarly R_w, can equivalently be written in the following manner using sets $U, U^c, W, W^c, V,$ and V^c.

$$R_u = \bigoplus_{U \cap V} u_k v_k \bigoplus_{U^c \cap V} u_k v_k \bigoplus_{U \cap V^c} u_k v_k \bigoplus_{U^c \cap V^c} u_k v_k$$

Each of the above sets above can be further decomposed with respect to W. In particular, we can write $U \cap V$ as $(U \cap W \cap V) \cup (U \cap W^c \cap V)$, and similarly for the other sets.

We show derivations of inequalities with $|U \cap V|$ and $|W \cap V|$ even in Theorems 4.15-4.18. Derivations for inequalities based on other permutations of $|U \cap V|, |W \cap V| \in \{\text{odd, even}\}$ are analogous.

We let $\mathcal{I}'_{uw} = \mathcal{I}_{uw} \cap \{(u, w, v, z_u, z_w) | R_u = 0, R_w = 0\}$. The dimension of $\mathcal{I}'_{uw}$ is 8 by Proposition 4.19. This follows because $r = 2$ after fixing R_u and R_w to 0.

1. **Family 1:** Consider the following inequality.

$$(1 - w_j) + v_i + (1 - z_u) + z_w \geq 1 \tag{4.50}$$

It is easy to see that this is a valid inequality for $\mathcal{I}'_{uw}$ since it is only violated when left-hand-side is 0, i.e., $w_j = 1, v_i = 0, z_u = 1, z_w = 0$. Fixing these in (4.26) results in the following system which is clearly infeasible:

$$\overset{1}{\cancel{z_u}} = u_i \overset{0}{\cancel{v_i}} \oplus u_j v_j$$
$$\overset{0}{\cancel{z_w}} = w_i \overset{0}{\cancel{v_i}} \oplus \overset{1}{\cancel{w_j}} v_j.$$

We next show that inequality (4.50) is facet defining for $\mathrm{conv}(\mathcal{I}'_{uw})$. The dimension of face $F = \mathrm{conv}(\mathcal{I}'_{uw}) \cap \{(1 - w_j) + v_i + (1 - z_u) + z_w = 1\}$ is 7. We show this by constructing 8 affinely independent points $(u_i, u_j, w_i, w_j, v_i\, v_j, z_u, z_w)$ on F as follows:

$$\begin{aligned}
&(0,1,0,1,0,1,1,1), \qquad (0,1,1,1,0,1,1,1),\\
&(1,1,0,1,0,1,1,1), \qquad (0,0,0,1,1,1,1,0),\\
&(0,0,0,1,0,0,0,0), \qquad (0,1,0,1,0,0,0,0),\\
&(0,0,0,1,1,0,1,0), \qquad (0,1,0,0,0,1,1,0).
\end{aligned}$$

We now sequentially lift the inequality (4.50) to obtain a facet defining inequality for $\mathrm{conv}(\mathcal{I}_{uw})$. In particular, we lift the following inequality:

$$(1 - w_j) + v_i + (1 - z_u) + z_w + \sum_{k \in U} \alpha_k (1 - u_k) + \sum_{k \in U^c} \alpha_k u_k + \sum_{k \in W} \beta_k (1 - w_k) + \sum_{k \in W^c} \beta_k w_k + \sum_{k \in V} \gamma_k (1 - v_k) + \sum_{k \in V^c} \gamma_k v_k \geq 1. \quad (4.52)$$

We lift the inequality sequentially in the following order: $U \cap W \cap V, U \cap W^c \cap V, U^c \cap W \cap V, U^c \cap W^c \cap V, U \cap W \cap V^c, U^c \cap W \cap V^c, U \cap W^c \cap V^c, U^c \cap W^c \cap V^c$. For each set, we first lift u variables, then w variables, and finally v variables.

- Set $U \cap W \cap V = \{1, 2, \ldots, l, l+1, \ldots, |U \cap W \cap V|\}$:

$$\begin{aligned}
\alpha_{l+1} = 1 - \max \Big\{ & 1 - w_j + v_i + 1 - z_u + z_w + \sum_{k=1}^{l} (1 - u_k) \mid (u, w, v, z) \in \mathcal{I}_{uw}, \\
& u_{l+1} = 0, u_{U \setminus [l+1]} = w_W = v_V = 1, u_{U^c} = w_{W^c} = v_{V^c} = 0 \Big\} \\
= 1,
\end{aligned}$$

$$\beta_{l+1} = 1 - \max\Big\{1 - w_j + v_i + 1 - z_u + z_w + \sum_{U\cap W\cap V}(1-u_k) + \sum_{k=1}^{l}(1-w_k) \mid (u,w,v,z) \in \mathcal{I}_{uw}, w_{l+1} = 0, u_{U\setminus U\cap W\cap V} = w_{W\setminus[l+1]} = v_V = 1, u_{U^c} = w_{W^c} = v_{V^c} = 0\Big\}$$
$$= 1,$$
$$\gamma_{l+1} = 1 - \max\Big\{1 - w_j + v_i + 1 - z_u + z_w + \sum_{U\cap W\cap V}(1-u_k+1-w_k) + \sum_{k=1}^{l}(1-v_k) \mid (u,w,v,z) \in \mathcal{I}_{uw}, v_{l+1} = 0, u_{U\setminus U\cap W\cap V} = 1, w_{W\setminus U\cap W\cap V} = v_{V\setminus[l]} = 1, u_{U^c} = w_{W^c} = v_{V^c} = 0\Big\}$$
$$= 1.$$

- Set $U \cap W^c \cap V = \{1, 2, \ldots, l, l+1, \ldots, |U \cap W^c \cap V|\}$:

$$\alpha_{l+1} = 1 - \max\Big\{1 - w_j + v_i + 1 - z_u + z_w + \sum_{U\cap W\cap V}(1-u_k+1-w_k+1-v_k) + \sum_{k=1}^{l}(1-u_k) \mid (u,w,v,z) \in \mathcal{I}_{uw}, u_{l+1} = 0, u_{U\setminus U\cap W\cap V\setminus[l+1]} = 1, w_{W\setminus U\cap W\cap V} = v_{V\setminus U\cap W\cap V} = 1, u_{U^c} = w_{W^c} = v_{V^c} = 0\Big\}$$
$$= 1,$$
$$\beta_{l+1} = 1 - \max\Big\{1 - w_j + v_i + 1 - z_u + z_w + \sum_{U\cap W\cap V}(1-u_k+1-w_k+1-v_k) + \sum_{U\cap W^c\cap V}(1-u_k) + \sum_{k=1}^{l} w_k \mid (u,w,v,z) \in \mathcal{I}_{uw}, w_{l+1} = 1, u_{U\setminus U\cap V} = 1, w_{W\setminus U\cap W\cap V} = v_{V\setminus U\cap W\cap V} = 1, u_{U^c} = 0, w_{W^c\setminus[l+1]} = v_{V^c} = 0\Big\}$$
$$= 1,$$
$$\gamma_{l+1} = 1 - \max\Big\{1 - w_j + v_i + 1 - z_u + z_w + \sum_{U\cap W\cap V}(1-u_k+1-w_k+1-v_k) + \sum_{U\cap W^c\cap V}(1-u_k+w_k) + \sum_{k=1}^{l}(1-v_k) \mid (u,w,v,z) \in \mathcal{I}_{uw}, v_{l+1} = 0, u_{U\setminus U\cap V} = 1, w_{W\setminus U\cap W\cap V} = v_{V\setminus U\cap W\cap V\setminus[l+1]} = 1,$$

$$u_{U^c} = w_{W^c \setminus U \cap W^c \cap V} = v_{V^c} = 0 \Big\}$$

$$= 1$$

- Set $U^c \cap W \cap V = \{1, 2, \ldots, l, l+1, \ldots, |U^c \cap W \cap V|\}$

$$\alpha_{l+1} = 1 - \max \Big\{ 1 - w_j + v_i + 1 - z_u + z_w + \sum_{U \cap W \cap V} (3 - u_k - w_k - v_k) +$$
$$\sum_{U \cap W^c \cap V} (2 - u_k + w_k - v_k) + \sum_{k=1}^{l} u_k \mid (u, w, v, z) \in \mathcal{I}_{uw},$$
$$u_{l+1} = 1, u_{U \setminus U \cap V} = 1, w_{W \setminus U \cap W \cap V} = v_{V \setminus U \cap V} = 1,$$
$$u_{U^c \setminus [l+1]} = w_{W^c \setminus U \cap W^c \cap V} = v_{V^c} = 0 \Big\}$$

$$= 1$$

$$\beta_{l+1} = 1 - \max \Big\{ 1 - w_j + v_i + 1 - z_u + z_w + \sum_{U \cap W \cap V} (3 - u_k - w_k - v_k) +$$
$$\sum_{U \cap W^c \cap V} (2 - u_k + w_k - v_k) + \sum_{U^c \cap W \cap V} u_k + \sum_{[l]} (1 - w_k) \mid$$
$$(u, w, v, z) \in \mathcal{I}_{uw}, w_{l+1} = 0, u_{U \setminus U \cap V} = 1, w_{W \setminus U \cap W \cap V \setminus [l+1]} = 1,$$
$$v_{V \setminus U \cap V} = 1, u_{U^c \setminus U^c \cap W \cap V} = w_{W^c \setminus U \cap W^c \cap V} = v_{V^c} = 0 \Big\}$$

$$= 1$$

$$\gamma_{l+1} = 1 - \max \Big\{ 1 - w_j + v_i + 1 - z_u + z_w + \sum_{U \cap W \cap V} (3 - u_k - w_k - v_k) +$$
$$\sum_{U \cap W^c \cap V} (2 - u_k + w_k - v_k) + \sum_{U^c \cap W \cap V} (1 + u_k - w_k) +$$
$$\sum_{[l]} (1 - v_k) \mid (u, w, v, z) \in \mathcal{I}_{uw}, v_{l+1} = 0, u_{U \setminus U \cap V} = 1,$$
$$w_{W \setminus W \cap V} = 1, v_{V \setminus U \cap V \setminus [l+1]} = 1, u_{U^c \setminus U^c \cap W \cap V} = 0,$$
$$w_{W^c \setminus U \cap W^c \cap V} = v_{V^c} = 0 \Big\}$$

$$= 1$$

- Set $U^c \cap W^c \cap V = \{1, 2, \ldots, l, l+1, \ldots, |U^c \cap W^c \cap V|\}$

$$\alpha_{l+1} = 1 - \max \Big\{ 1 - w_j + v_i + 1 - z_u + z_w + \sum_{U \cap W \cap V} (3 - u_k - w_k - v_k) +$$

$$\sum_{U\cap W^c\cap V}(2-u_k+w_k-v_k)+\sum_{U^c\cap W\cap V}(2+u_k-w_k-v_k)+$$
$$\sum_{[l]}u_k \mid (u,w,v,z)\in\mathcal{I}_{uw}, u_{l+1}=1, u_{U\setminus U\cap V}=1, w_{W\setminus W\cap V}=1,$$
$$v_{V\setminus(U\cup W)\cap V}=1, u_{U^c\setminus U^c\cap W\cap V\setminus[l+1]}=w_{W^c\setminus U\cap W^c\cap V}=v_{V^c}=0\Big\}$$
$$=1$$

$$\beta_{l+1}=1-\max\Big\{1-w_j+v_i+1-z_u+z_w+\sum_{U\cap W\cap V}(3-u_k-w_k-v_k)+$$
$$\sum_{U\cap W^c\cap V}(2-u_k+w_k-v_k)+\sum_{U^c\cap W\cap V}(2+u_k-w_k-v_k)+$$
$$\sum_{U^c\cap W^c\cap V}u_k+\sum_{[l]}w_k \mid (u,w,v,z)\in\mathcal{I}_{uw}, w_{l+1}=1, u_{U\setminus U\cap V}=1,$$
$$w_{W\setminus W\cap V}=1, v_{V\setminus(U\cup W)\cap V}=1, u_{U^c\setminus U^c\cap V}=0$$
$$w_{W^c\setminus U\cap W^c\cap V\setminus[l+1]}=0, v_{V^c}=0\Big\}$$
$$=1$$

$$\gamma_{l+1}=1-\max\Big\{1-w_j+v_i+1-z_u+z_w+\sum_{U\cap W\cap V}(3-u_k-w_k-v_k)+$$
$$\sum_{U\cap W^c\cap V}(2-u_k+w_k-v_k)+\sum_{U^c\cap W\cap V}(2+u_k-w_k-v_k)+$$
$$\sum_{U^c\cap W^c\cap V}(u_k+w_k) \mid (u,w,v,z)\in\mathcal{I}_{uw}, v_{l+1}=0, u_{U\setminus U\cap V}=1,$$
$$w_{W\setminus W\cap V}=1, v_{V\setminus[l+1]\setminus(U\cup W)\cap V}=1, u_{U^c\setminus U^c\cap V}=0$$
$$w_{W^c\setminus W^c\cap V}=0, v_{V^c}=0\Big\}$$
$$=0$$

- Set $U\cap W\cap V^c=\{1,2,\ldots,l,l+1,\ldots,|U\cap W\cap V^c|\}$

$$\alpha_{l+1}=1-\max\Big\{1-w_j+v_i+1-z_u+z_w+\sum_{U\cap W\cap V}(3-u_k-w_k-v_k)+$$
$$\sum_{U\cap W^c\cap V}(2-u_k+w_k-v_k)+\sum_{U^c\cap W\cap V}(2+u_k-w_k-v_k)+$$
$$\sum_{U^c\cap W^c\cap V}(u_k+w_k) \mid (u,w,v,z)\in\mathcal{I}_{uw}, u_{l+1}=0,$$
$$u_{U\setminus U\cap V\setminus[l+1]}=1, w_{W\setminus W\cap V}=1, u_{U^c\setminus U^c\cap V}=w_{W^c\setminus W^c\cap V}=0,$$
$$v_{V^c}=0\Big\}$$

$$= 0$$

$$\beta_{l+1} = 1 - \max\Big\{1 - w_j + v_i + 1 - z_u + z_w + \sum_{U\cap W\cap V}(3 - u_k - w_k - v_k) + \sum_{U\cap W^c\cap V}(2 - u_k + w_k - v_k) + \sum_{U^c\cap W\cap V}(2 + u_k - w_k - v_k) + \sum_{U^c\cap W^c\cap V}(u_k + w_k) \mid (u,w,v,z) \in \mathcal{I}_{uw}, w_{l+1} = 0, u_{U\cap V^c\setminus U\cap W\cap V^c} = 1, w_{W\setminus W\cap V\setminus[l+1]} = 1, u_{U^c\setminus U^c\cap V} = 0, w_{W^c\setminus W^c\cap V} = 0, v_{V^c} = 0\Big\}$$

$$= 0$$

$$\gamma_{l+1} = 1 - \max\Big\{1 - w_j + v_i + 1 - z_u + z_w + \sum_{U\cap W\cap V}(3 - u_k - w_k - v_k) + \sum_{U\cap W^c\cap V}(2 - u_k + w_k - v_k) + \sum_{U^c\cap W\cap V}(2 + u_k - w_k - v_k) + \sum_{U^c\cap W^c\cap V}(u_k + w_k) + \sum_{k=1}^{l} v_k \mid (u,w,v,z) \in \mathcal{I}_{uw}, v_{l+1} = 1, u_{U\cap V^c\setminus U\cap W\cap V^c} = 1, w_{W\cap V^c\setminus U\cap W\cap V^c} = 1, u_{U^c\setminus U^c\cap V} = 0, w_{W^c\setminus W^c\cap V} = 0, v_{V^c\setminus[l+1]} = 0\Big\}$$

$$= 1$$

- Set $U \cap W^c \cap V^c = \{1, 2, \ldots, l, l+1, \ldots, |U \cap W^c \cap V^c|\}$

$$\alpha_{l+1} = 1 - \max\Big\{1 - w_j + v_i + 1 - z_u + z_w + \sum_{U\cap W\cap V}(3 - u_k - w_k - v_k) + \sum_{U\cap W^c\cap V}(2 - u_k + w_k - v_k) + \sum_{U^c\cap W\cap V}(2 + u_k - w_k - v_k) + \sum_{U^c\cap W^c\cap V}(u_k + w_k) + \sum_{U\cap W\cap V^c} v_k \mid (u,w,v,z) \in \mathcal{I}_{uw}, u_{l+1} = 0, u_{U\cap W^c\cap V^c\setminus[l+1]} = 1, w_{U^c\cap W\cap V^c} = 1, u_{U^c\cap V^c} = 0, w_{W^c\cap V^c} = 0, v_{V^c\setminus U\cap W\cap V^c} = 0\Big\}$$

$$= 0$$

$$\beta_{l+1} = 1 - \max\Big\{1 - w_j + v_i + 1 - z_u + z_w + \sum_{U\cap W\cap V}(3 - u_k - w_k - v_k) + \sum_{U\cap W^c\cap V}(2 - u_k + w_k - v_k) + \sum_{U^c\cap W\cap V}(2 + u_k - w_k - v_k) +$$

$$\sum_{U^c\cap W^c\cap V}(u_k+w_k)+\sum_{U\cap W\cap V^c}v_k \mid (u,w,v,z)\in\mathcal{I}_{uw}, w_{l+1}=1,$$
$$w_{U^c\cap W\cap V^c}=1, u_{U^c\cap V^c}=0, w_{W^c\cap V^c\setminus[l+1]}=0, v_{V^c\setminus U\cap W\cap V^c}=0\Big\}$$
$$=0$$

$$\gamma_{l+1}=1-\max\Big\{1-w_j+v_i+1-z_u+z_w+\sum_{U\cap W\cap V}(3-u_k-w_k-v_k)+$$
$$\sum_{U\cap W^c\cap V}(2-u_k+w_k-v_k)+\sum_{U^c\cap W\cap V}(2+u_k-w_k-v_k)+$$
$$\sum_{U^c\cap W^c\cap V}(u_k+w_k)+\sum_{U\cap W\cap V^c}v_k+\sum_{k=1}^{l}v_k \mid (u,w,v,z)\in\mathcal{I}_{uw},$$
$$v_{l+1}=1, w_{U^c\cap W\cap V^c}=1, u_{U^c\cap V^c}=0, w_{W^c\cap V^c\setminus U\cap W^c\cap V^c}=0,$$
$$v_{V^c\setminus U\cap W\cap V^c\setminus[l+1]}=0\Big\}$$
$$=1$$

- Set $U^c\cap W\cap V^c=\{1,2,\ldots,l,l+1,\ldots,|U^c\cap W\cap V^c|\}$

$$\alpha_{l+1}=1-\max\Big\{1-w_j+v_i+1-z_u+z_w+\sum_{U\cap W\cap V}(3-u_k-w_k-v_k)+$$
$$\sum_{U\cap W^c\cap V}(2-u_k+w_k-v_k)+\sum_{U^c\cap W\cap V}(2+u_k-w_k-v_k)+$$
$$\sum_{U^c\cap W^c\cap V}(u_k+w_k)+\sum_{U\cap W\cap V^c}v_k+\sum_{U\cap W^c\cap V^c}v_k \mid$$
$$(u,w,v,z)\in\mathcal{I}_{uw}, u_{l+1}=1, w_{U^c\cap W\cap V^c}=1, u_{U^c\cap V^c\setminus[l+1]}=0,$$
$$w_{U^c\cap W^c\cap V^c}=0, v_{V^c\setminus U\cap V^c}=0\Big\}$$
$$=0$$

$$\beta_{l+1}=1-\max\Big\{1-w_j+v_i+1-z_u+z_w+\sum_{U\cap W\cap V}(3-u_k-w_k-v_k)+$$
$$\sum_{U\cap W^c\cap V}(2-u_k+w_k-v_k)+\sum_{U^c\cap W\cap V}(2+u_k-w_k-v_k)+$$
$$\sum_{U^c\cap W^c\cap V}(u_k+w_k)+\sum_{U\cap W\cap V^c}v_k+\sum_{U\cap W^c\cap V^c}v_k \mid$$
$$(u,w,v,z)\in\mathcal{I}_{uw}, w_{l+1}=0, w_{U^c\cap W\cap V^c\setminus[l+1]}=1,$$
$$u_{U^c\cap W^c\cap V^c}=0, w_{U^c\cap W^c\cap V^c}=0, v_{U^c\cap V^c}=0\Big\}$$
$$=0$$

$$\gamma_{l+1} = 1 - \max\Big\{1 - w_j + v_i + 1 - z_u + z_w + \sum_{U\cap W\cap V}(3 - u_k - w_k - v_k) + \sum_{U\cap W^c\cap V}(2 - u_k + w_k - v_k) + \sum_{U^c\cap W\cap V}(2 + u_k - w_k - v_k) + \sum_{U^c\cap W^c\cap V}(u_k + w_k) + \sum_{U\cap W\cap V^c} v_k + \sum_{U\cap W^c\cap V^c} v_k + \sum_{k=1}^{l} v_k \mid (u,w,v,z) \in \mathcal{I}_{uw}, v_{l+1} = 1, u_{U^c\cap W^c\cap V^c} = 0, w_{U^c\cap W^c\cap V^c} = 0, v_{U^c\cap V^c\setminus[l+1]} = 0\Big\}$$

$$= 1$$

- Set $U^c \cap W^c \cap V^c = \{1, 2, \ldots, l, l+1, \ldots, |U^c \cap W^c \cap V^c|\}$

$$\alpha_{l+1} = 1 - \max\Big\{1 - w_j + v_i + 1 - z_u + z_w + \sum_{U\cap W\cap V}(3 - u_k - w_k - v_k) + \sum_{U\cap W^c\cap V}(2 - u_k + w_k - v_k) + \sum_{U^c\cap W\cap V}(2 + u_k - w_k - v_k) + \sum_{U^c\cap W^c\cap V}(u_k + w_k) + \sum_{(U\cup W)\cap V^c} v_k \mid (u,w,v,z) \in \mathcal{I}_{uw}, u_{l+1} = 1, u_{U^c\cap W^c\cap V^c\setminus[l+1]} = 0, w_{U^c\cap W^c\cap V^c} = 0, v_{U^c\cap W^c\cap V^c} = 0\Big\}$$

$$= 0$$

$$\beta_{l+1} = 1 - \max\Big\{1 - w_j + v_i + 1 - z_u + z_w + \sum_{U\cap W\cap V}(3 - u_k - w_k - v_k) + \sum_{U\cap W^c\cap V}(2 - u_k + w_k - v_k) + \sum_{U^c\cap W\cap V}(2 + u_k - w_k - v_k) + \sum_{U^c\cap W^c\cap V}(u_k + w_k) + \sum_{(U\cup W)\cap V^c} v_k \mid (u,w,v,z) \in \mathcal{I}_{uw}, w_{l+1} = 1, w_{U^c\cap W^c\cap V^c\setminus[l+1]} = 0, v_{U^c\cap W^c\cap V^c} = 0\Big\}$$

$$= 0$$

$$\gamma_{l+1} = 1 - \max\Big\{1 - w_j + v_i + 1 - z_u + z_w + \sum_{U\cap W\cap V}(3 - u_k - w_k - v_k) + \sum_{U\cap W^c\cap V}(2 - u_k + w_k - v_k) + \sum_{U^c\cap W\cap V}(2 + u_k - w_k - v_k) +$$

$$\sum_{U^c\cap W^c\cap V}(u_k+w_k)+\sum_{(U\cup W)\cap V^c}v_k \mid$$
$$(u,w,v,z)\in\mathcal{I}_{uw}, v_{l+1}=1,$$
$$v_{U^c\cap W^c\cap V^c\setminus[l+1]}=0\Big\}$$
$$=0$$

We summarize the lifting coefficients obtained via sequential lifting as follows:

$$\begin{aligned}
&\alpha_k=1,\beta_k=1,\gamma_k=1 \quad \forall k\in U\cap W\cap V, U\cap W^c\cap V, U^c\cap W\cap V\\
&\alpha_k=1,\beta_k=1,\gamma_k=0 \quad \forall k\in U^c\cap W^c\cap V\\
&\alpha_k=0,\beta_k=0,\gamma_k=1 \quad \forall k\in U\cap W\cap V^c, U^c\cap W\cap V^c, U\cap W^c\cap V^c\\
&\alpha_k=0,\beta_k=0,\gamma_k=0 \quad \forall k\in U^c\cap W^c\cap V^c.
\end{aligned}$$

Assuming $U^c\cap W^c\cap V^c=\emptyset$ results in inequality (4.27).

2. **Family 2:** Consider the following inequality.

$$(1-u_i)+(1-u_j)+(1-w_i)+(1-w_j)+(1-z_u)+z_w\geq 1. \tag{4.53}$$

It is easy to see that this is a valid inequality for $\mathcal{I}'_{uw}$ since it is only violated when left-hand-side is 0, i.e., $u_i=1,u_j=1,w_i=1,w_j=1,z_u=1,z_w=0$. Fixing these in (4.26) results in the following system which is clearly infeasible:

$$\begin{aligned}
\overset{1}{\cancel{z_u}}&=\overset{1}{\cancel{w_i}}v_i\oplus\overset{1}{\cancel{w_j}}v_j\\
\overset{0}{\cancel{z_w}}&=\overset{1}{\cancel{w_i}}v_i\oplus\overset{1}{\cancel{w_j}}v_j.
\end{aligned}$$

We next show that inequality (4.53) is facet defining for $\mathrm{conv}(\mathcal{I}'_{uw})$. The dimension of face $F=\mathrm{conv}(\mathcal{I}'_{uw})\cap\{(1-u_i)+(1-u_j)+(1-w_i)+(1-w_j)+(1-z_u)+z_w=1\}$ is 7.

Similar to the previous case, we construct 8 points $(u_i,u_j,w_i,w_j,v_i\,v_j,z_u,z_w)$ on F which are affinely independent as follows:

$$\begin{aligned}
&(1,1,1,1,0,1,1,1), \qquad (1,1,1,1,1,0,1,1),\\
&(1,1,1,1,0,0,0,0), \qquad (1,1,1,1,1,1,0,0),\\
&(1,1,1,0,0,1,1,0), \qquad (1,1,0,1,1,0,1,0),\\
&(1,0,1,1,1,1,1,0), \qquad (0,1,1,1,1,1,1,0).
\end{aligned}$$

We now sequentially lift the inequality (4.53) to obtain a facet defining inequality for $\text{conv}(\mathcal{I}_{uw})$. In particular, we lift the following inequality:

$$(1-u_i)+(1-u_j)+(1-w_i)+(1-w_j)+(1-z_u)+z_w+\sum_{k\in U}\alpha_k(1-u_k)+\sum_{k\in U^c}\alpha_k u_k+$$
$$\sum_{k\in W}\beta_k(1-w_k)+\sum_{k\in W^c}\beta_k w_k+\sum_{k\in V}\gamma_k(1-v_k)+\sum_{k\in V^c}\gamma_k v_k\geq 1. \quad (4.55)$$

Sequential lifting, similar to Family 1, gives the following coefficients:

$$\begin{aligned}
&\alpha_k=1,\beta_k=1,\gamma_k=0 \quad \forall k\in U\cap W\cap V, U^c\cap W^c\cap V\\
&\alpha_k=1,\beta_k=1,\gamma_k=1 \quad \forall k\in U\cap W^c\cap V, U^c\cap W\cap V\\
&\alpha_k=0,\beta_k=0,\gamma_k=1 \quad \forall k\in U\cap W^c\cap V^c, U^c\cap W\cap V^c\\
&\alpha_k=0,\beta_k=0,\gamma_k=0 \quad \forall k\in U\cap W\cap V^c, U^c\cap W^c\cap V^c.
\end{aligned}$$

Assuming $U\cap W\cap V^c, U^c\cap W^c\cap V^c=\emptyset$ results in inequality (4.31).

3. **Family 3:** Consider the following inequality.

$$(1-u_i)+(1-u_j)+v_i+v_j+z_u+2(1-z_w)\geq 2. \quad (4.56)$$

It is easy to see that this is a valid inequality for $\mathcal{I}'_{uw}$ since it is only violated when left-hand-side $\in\{0,1\}$. Left-hand-side is 0 if $u_i=1,u_j=1,v_i=0,v_j=0,z_u=0,z_w=1$. Fixing these in (4.26) results in the following system which is clearly infeasible:

$$\begin{aligned}
\overset{0}{\cancel{z_u}} &= \overset{1}{\cancel{u_i}}\overset{0}{\cancel{v_i}}\oplus\overset{1}{\cancel{u_j}}\overset{0}{\cancel{v_j}}\\
\overset{1}{\cancel{z_w}} &= w_i\overset{0}{\cancel{v_i}}\oplus w_j\overset{0}{\cancel{v_j}}.
\end{aligned}$$

Left-hand-side is 1 if exactly one of the above fixed variables except z_w switches. Switching u_i,u_j to 0 does not fix infeasibility since the second equation in the above system still remains infeasible. Switching a v_i or v_j variable to 1 fixes infeasibility for the second equation but makes the first infeasible, thus leave the entire system still infeasible. We next show that inequality (4.56) is facet defining for $\text{conv}(\mathcal{I}'_{uw})$. The dimension of face $F=\text{conv}(\mathcal{I}'_{uw})\cap\{(1-u_i)+(1-u_j)+v_i+v_j+z_u+2(1-z_w)=2\}$ is 7. Similar to the previous case, we construct 8 points $(u_i,u_j,w_i,w_j,v_i\,v_j,z_u,z_w)$ on

F which are affinely independent as follows:

$$\begin{aligned}
&(1,1,0,0,0,0,0,0), \qquad (1,0,0,1,0,1,0,1),\\
&(1,1,0,1,0,0,0,0), \qquad (1,1,1,0,1,0,1,1),\\
&(1,1,1,0,0,0,0,0), \qquad (1,1,0,1,0,1,1,1),\\
&(0,1,1,0,1,0,0,1), \qquad (1,1,1,0,1,1,0,1).
\end{aligned}$$

We now sequentially lift the inequality (4.56) to obtain a facet defining inequality for $\mathrm{conv}(\mathcal{I}_{uw})$. In particular, we lift the following inequality:

$$(1-u_i)+(1-u_j)+v_i+v_j+z_u+2(1-z_w)+\sum_{k\in U}\alpha_k(1-u_k)+\sum_{k\in U^c}\alpha_k u_k+ \\ \sum_{k\in W}\beta_k(1-w_k)+\sum_{k\in W^c}\beta_k w_k+\sum_{k\in V}\gamma_k(1-v_k)+\sum_{k\in V^c}\gamma_k v_k\geq 2. \quad (4.58)$$

Sequential lifting, similar to Family 1, gives the following coefficients:

$$\begin{aligned}
&\alpha_k=1,\beta_k=2,\gamma_k=1 \quad \forall k\in U\cap W\cap V\\
&\alpha_k=1,\beta_k=2,\gamma_k=1 \quad \forall k\in U\cap W^c\cap V\\
&\alpha_k=1,\beta_k=2,\gamma_k=2 \quad \forall k\in U^c\cap W\cap V\\
&\alpha_k=1,\beta_k=2,\gamma_k=0 \quad \forall k\in U^c\cap W^c\cap V\\
&\alpha_k=0,\beta_k=0,\gamma_k=2 \quad \forall k\in U^c\cap W\cap V^c\\
&\alpha_k=0,\beta_k=0,\gamma_k=1 \quad \forall k\in U\cap V^c\\
&\alpha_k=0,\beta_k=0,\gamma_k=0 \quad \forall k\in U^c\cap W^c\cap V^c.
\end{aligned}$$

Assuming $U\cap V^c, U^c\cap W^c\cap V^c=\emptyset$ results in inequality (4.35).

4. **Family 4:** Consider the following inequality:

$$(1-u_i)+(1-u_j)+2(1-w_i)+(1-v_i)+v_j+(1-z_u)+2z_w\geq 2. \quad (4.59)$$

It is easy to see that this is a valid inequality for $\mathcal{I}'_{uw}$ since it is only violated when left-hand-side $\in\{0,1\}$. Left-hand-side is 0 if $u_i=1,u_j=1,w_i=1,v_i=1,v_j=0,z_u=1,z_w=0$. Fixing these in (4.26) results in the following system which is

clearly infeasible:

$$\overset{1}{\cancel{z_u}} = \overset{1}{\cancel{u_i}}\overset{1}{\cancel{v_i}} \oplus \overset{1}{\cancel{u_j}}\overset{0}{\cancel{v_j}}$$
$$\overset{0}{\cancel{z_w}} = \overset{1}{\cancel{w_i}}\overset{1}{\cancel{v_i}} \oplus w_j\overset{0}{\cancel{v_j}}.$$

Left-hand-side is 1 if exactly one of the above fixed variables exceptw_i, z_w switches. Switching u_i, u_j to 0 does not fix infeasibility since the second equation in the above system still remains infeasible. Switching a v_i to 1 or v_j variable to 0 fixes infeasibility for the second equation but makes the first infeasible, thus leave the entire system still infeasible. The dimension of face $F = \text{conv}(\mathcal{I}'_{uw}) \cap \{(1-u_i)+(1-u_j)+2(1-w_i)+(1-v_i)+v_j+(1-z_u)+2z_w = 2\}$ is 7. Similar to the previous case, we construct 8 points $(u_i, u_j, w_i, w_j, v_i\, v_j, z_u, z_w)$ on F which are affinely independent as follows:

$$\begin{aligned}
&(1,1,1,0,1,0,1,1), \qquad &&(1,0,1,1,1,1,1,0),\\
&(1,1,1,1,1,0,1,1) \qquad &&(1,1,1,0,0,0,0,0),\\
&(1,1,0,0,1,0,1,0) \qquad &&(1,1,1,0,1,1,0,0),\\
&(0,1,1,1,1,1,1,0) \qquad &&(0,1,1,0,1,0,0,0),
\end{aligned}$$

We now sequentially lift the inequality (4.59) to obtain a facet defining inequality for $\text{conv}(\mathcal{I}_{uw})$. In particular, we lift the following inequality:

$$(1-u_i)+(1-u_j)+2(1-w_i)+(1-v_i)+v_j+(1-z_u)+2z_w+\sum_{k\in U}\alpha_k(1-u_k)+\sum_{k\in U^c}\alpha_k u_k+$$
$$\sum_{k\in W}\beta_k(1-w_k)+\sum_{k\in W^c}\beta_k w_k+\sum_{k\in V}\gamma_k(1-v_k)+\sum_{k\in V^c}\gamma_k v_k \geq 2. \quad (4.61)$$

Sequential lifting, similar to Family 1, gives the following coefficients:

$$\begin{aligned}
&\alpha_k = 1, \beta_k = 2, \gamma_k = 1 \quad \forall k \in U \cap W \cap V\\
&\alpha_k = 1, \beta_k = 2, \gamma_k = 1 \quad \forall k \in U \cap W^c \cap V\\
&\alpha_k = 1, \beta_k = 2, \gamma_k = 2 \quad \forall k \in U^c \cap W \cap V\\
&\alpha_k = 1, \beta_k = 2, \gamma_k = 0 \quad \forall k \in U^c \cap W^c \cap V\\
&\alpha_k = 0, \beta_k = 0, \gamma_k = 2 \quad \forall k \in U^c \cap W \cap V^c\\
&\alpha_k = 0, \beta_k = 0, \gamma_k = 1 \quad \forall k \in U \cap V^c\\
&\alpha_k = 0, \beta_k = 0, \gamma_k = 0 \quad \forall k \in U^c \cap W^c \cap V^c.
\end{aligned}$$

Assuming $U \cap V^c, U^c \cap W^c \cap V^c = \emptyset$ results in inequality (4.42).

Conjecture 4.20. *Lifting coefficients in sequential lifting of inequalities* (4.50), (4.53) (4.56), (4.59) *are order independent.*

Separating valid inequalities

We now discuss separation procedure for the inequalities proposed in Section 4.4.1. In particular, given a fractional solution $(\hat{u}, \hat{v}, \hat{z})$ to the compact formulation (4.13), we discuss methods that can determine if the solution is violated by any of the inequalities proposed in Section 4.4.1. We use the same notation as in Section 4.4.1 and refer to two rows of matrix $\hat{U}$ as $\hat{u}, \hat{w}$, column of $\hat{V}$ as $\hat{v}$, and corresponding $\hat{z}$ as $\hat{z}_u$ and $\hat{z}_w$. We discuss exact separation which involves solving a small integer program in Section 4.4.3 and a heuristic cut separation strategy in Section 4.4.4. We point out that there are $\mathcal{O}(d^2n + dn^2)$ choices for choosing $(\hat{u}, \hat{w}, \hat{v})$ pairs since one could either choose two rows of U and one column of V or one row of U and two columns of V. For each of these choices, we next discuss how to determine if there is a violated inequality for a given partial solution $(\hat{u}, \hat{w}, \hat{v}, \hat{z}_u, \hat{z}_w)$.

In exact separation, we let indices (i, j) be part of the partition, i.e., we generate at most one valid inequality of each family for a given $(\hat{u}, \hat{w}, \hat{v})$. On the other hand, in heuristic separation strategy, we iterate over all (i, j) pairs which can lead to $\mathcal{O}(r^2)$ inequalities in worst case.

4.4.3 Exact Separation

In order to separate a valid inequality, we minimize left-hand-side for a given $(\hat{u}, \hat{w}, \hat{v}, \hat{z}_u, \hat{z}_w)$ for each of the inequalities proposed in Section 4.4.1. We observe that to minimize left-hand-side each of the valid inequalities proposed in Theorems 4.15-4.18, we need to partition set $[r]$ in 7 different sets, say $\mathcal{S} = (S_1, S_2, \ldots, S_7)$. We use sets S_1 and S_2 to map to i and j respectively, and constrain them to be singletons. We always map S_3 to $U \cap W \cap V$, S_4 to $U \cap W^c \cap V$, and S_5 to $U^c \cap W \cap V$. Mapping of sets S_6 and S_7 varies for each family and will be described later. Since each family has different coefficients for partitioning, we next define a general integer program for partitioning $[r]$ while ensuring that two of the sets are singletons, to identify i and j, and also enforcing necessary cardinality constraints. We let $\mathbf{c} = [c_1, c_2, \ldots, c_7]$ where $c_j \in \mathbb{R}^r \; \forall j = 1, \ldots, 7$. We define the coefficients corresponding to each family in Table 4.1. We further let $\alpha = 1$ if we require $|U \cap V|$ to be odd and 0 otherwise. Similarly, we let $\beta = 1$ if we require $|W \cap V|$ to be odd and 0 otherwise.

		Family 1	Family 2	Family 3	Family 4
(i)	c_1	$\hat{v}$	$\mathbf{2} - \hat{u} - \hat{w}$	$\mathbf{1} - \hat{u} + \hat{v}$	$\mathbf{4} - \hat{u} - 2\hat{w} - \hat{v}$
(j)	c_2	$\mathbf{1} - \hat{w}$	$\mathbf{2} - \hat{u} - \hat{w}$	$\mathbf{1} - \hat{u} + \hat{v}$	$\mathbf{1} - \hat{u} + \hat{v}$
$(U \cap W \cap V)$	c_3	$\mathbf{3} - \hat{u} - \hat{w} - \hat{v}$	$\mathbf{2} - \hat{u} - \hat{w}$	$\mathbf{4} - \hat{u} - 2\hat{w} - \hat{v}$	$\mathbf{4} - \hat{u} - 2\hat{w} - \hat{v}$
$(U \cap W^c \cap V)$	c_4	$\mathbf{2} - \hat{u} - \hat{v} + \hat{w}$	$\mathbf{2} - \hat{u} + \hat{w} - \hat{v}$	$\mathbf{2} - \hat{u} + 2\hat{w} - \hat{v}$	$\mathbf{2} - \hat{u} + 2\hat{w} - \hat{v}$
$(U^c \cap W \cap V)$	c_5	$\mathbf{2} + \hat{u} - \hat{w} - \hat{v}$	$\mathbf{2} + \hat{u} - \hat{w} - \hat{v}$	$\mathbf{4} + \hat{u} - 2\hat{w} - 2\hat{v}$	$\mathbf{4} + \hat{u} - 2\hat{w} - 2\hat{v}$
-	c_6	$\hat{u} + \hat{w}$	$\hat{u} + \hat{w}$	$\hat{u} + 2\hat{w}$	$\hat{u} + 2\hat{w}$,
-	c_7	$\hat{v}$	$\hat{v}$	$2\hat{v}$	$2\hat{v}$

Table 4.1: Coefficient calculation for valid inequalities exact separation

For a given $\mathbf{c} \in \mathbb{R}^{r\times 7}$, $\alpha \in \{0,1\}, \beta \in \{0,1\}$, the following integer program partitions set $[r]$ in 7 sets such that S_1 and S_2 are singleton, $(|S_3 \cup S_4|) \mod 2 = \alpha$, and $|S_3 \cup S_5| \mod 2 = \beta$. We let θ_j^l be a binary variable which takes a value of 1 if index $l \in [r]$ is assigned to set S_j $\forall j = 1, \ldots, 7$. We also let θ_j denote $[\theta_j^1, \theta_j^2, \ldots, \theta_j^r]$.

$$\hat{p}(\alpha, \beta, \mathbf{c}) = \min\ c_1^\top\theta_1 + c_2^\top\theta_2 + c_3^\top\theta_3 + c_4^\top\theta_4 + c_5^\top\theta_5 + c_6^\top\theta_6 + c_7^\top\theta_7 \tag{4.62a}$$

$$\sum_{l\in[r]} \theta_1^l = 1 \tag{4.62b}$$

$$\sum_{l\in[r]} \theta_2^l = 1 \tag{4.62c}$$

$$\sum_{j=1}^{7} \theta_j^l = 1 \quad \forall l \in [r] \tag{4.62d}$$

$$\sum_{l\in[r]} [\theta_3^l + \theta_4^l] - 2t = \alpha \tag{4.62e}$$

$$\sum_{l\in[r]} [\theta_3^l + \theta_5^l] - 2t' = \beta \tag{4.62f}$$

$$\theta \in \{0,1\}^{r\times 7}, t \in \mathbb{Z}^+, t' \in \mathbb{Z}^+ \tag{4.62g}$$

Constraint (4.62c) enforces that $|S_1| = 1$, constraint (4.62d) enforces that $|S_2| = 1$, and constraint (4.62e) enforces that each index $k \in [r]$ is assigned to exactly one set S_j. Constraints (4.62e), (4.62f) ensure that $(|S_3 \cup S_4|) \mod 2 = \alpha$ and $|S_3 \cup S_5| \mod 2 = \beta$ respectively.

We use solution $\hat{\theta}$ to construct partition $\hat{\mathcal{S}}$. In particular, $\hat{S}_j = \{k \mid \hat{\theta}_j^k = 1\}$ $\forall j = 3, \ldots, 7$. We further let $\hat{S}_1 = \{s_1\}$ and $\hat{S}_2 = \{s_2\}$ since S_1 and S_2 are singletons.

1. Family 1: For a given partial solution $(\hat{u}, \hat{w}, \hat{v}, \hat{z}_u, \hat{z}_w)$, we determine if there are indices $i, j \in [r]$ and sets U, U^c, W, W^c, V, V^c such that $U \cup U^c = W \cup W^c = V \cup V^c =$

$[r] \setminus \{i,j\}, U^c \cap W^c \cap V^c = \emptyset$, and the corresponding Family 1 inequality violates the partial solution $(\hat{u}, \hat{w}, \hat{v}, \hat{z}_u, \hat{z}_w)$.

To separate Family 1, we solve the integer program (4.62) for parameters $(\alpha, \beta) \in \{(0,0),(0,1),(1,0),(1,1)\}$ with $\mathbf{c} = [c_1, c_2, c_3, c_4, c_5, c_6, c_7]$ where $c_l \ \forall l \in \{1, 2, \dots 7\}$ corresponds to Family 1 in Table 4.1.

The solution to integer program (4.62) can be mapped to i, j and sets of interest U, U^c, W, W^c, V, V^c as follows.

$$\begin{aligned} &i = s_1, j = s_2, U \cap W \cap V = \hat{S}_3, U \cap W^c \cap V = \hat{S}_4 \\ &U^c \cap W \cap V = \hat{S}_5, U^c \cap W^c \cap V = \hat{S}_6, (U \cup W) \cap V^c = \hat{S}_7 \end{aligned} \tag{4.63}$$

- $\alpha = 0, \beta = 0$: If $\hat{p}(0, 0, \mathbf{c}) + (1 - \hat{z}_u) + \hat{z}_w < 1$, then we have found a violated inequality of type (4.27) for the $i, j, U, U^c, W, W^c, V, V^c$ constructed in (4.63).
- $\alpha = 0, \beta = 1$: If $\hat{p}(0, 1, \mathbf{c}) + (1 - \hat{z}_u) + 1 - \hat{z}_w < 1$, then we have found a violated inequality of type (4.28) for the $i, j, U, U^c, W, W^c, V, V^c$ constructed in (4.63).
- $\alpha = 1, \beta = 0$: If $\hat{p}(1, 0, \mathbf{c}) + \hat{z}_u + \hat{z}_w < 1$, then we have found a violated inequality of type (4.29) for the $i, j, U, U^c, W, W^c, V, V^c$ constructed in (4.63).
- $\alpha = 1, \beta = 1$: If $\hat{p}(1, 1, \mathbf{c}) + \hat{z}_u + (1 - \hat{z}_w) < 1$, then we have found a violated inequality of type (4.30) for the $i, j, U, U^c, W, W^c, V, V^c$ constructed in (4.63).

2. Family 2: For a given partial solution $(\hat{u}, \hat{w}, \hat{v}, \hat{z}_u, \hat{z}_w)$, we determine if there are indices $i, j \in [r]$ and sets U, U^c, W, W^c, V, V^c such that $U \cup U^c = W \cup W^c = V \cup V^c = [r] \setminus \{i, j\}, U^c \cap W^c \cap V^c = \emptyset, U \cap W \cap V^c = \emptyset$, and the corresponding Family 2 inequality violates the partial solution $(\hat{u}, \hat{w}, \hat{v}, \hat{z}_u, \hat{z}_w)$. To separate Family 2, we solve the integer program (4.62) for parameters $(\alpha, \beta) \in \{(0,0),(0,1),(1,0),(1,1)\}$ with with $\mathbf{c} = [c_1, c_2, c_3, c_4, c_5, c_6, c_7]$ where $c_l \ \forall l \in \{1, 2, \dots 7\}$ corresponds to Family 2 in Table 4.1. The solution to integer program (4.62) can be mapped to i, j and sets of interest U, U^c, W, W^c, V, V^c as follows.

$$\begin{aligned} &i = s_1, j = s_2, U \cap W \cap V = \hat{S}_3, U \cap W^c \cap V = \hat{S}_4 \\ &U^c \cap W \cap V = \hat{S}_5, U^c \cap W^c \cap V = \hat{S}_6, (U \triangle W) \cap V^c = \hat{S}_7 \end{aligned} \tag{4.64}$$

- $\alpha = 0, \beta = 0$: If $\hat{p}(0, 0, \mathbf{c}) + (1 - \hat{z}_u) + \hat{z}_w < 1$, then we have found a violated inequality of type (4.31) for the $i, j, U, U^c, W, W^c, V, V^c$ constructed in (4.64).
- $\alpha = 0, \beta = 1$: If $\hat{p}(0, 1, \mathbf{c}) + (1 - \hat{z}_u) + 1 - \hat{z}_w < 1$, then we have found a violated inequality of type (4.32) for the $i, j, U, U^c, W, W^c, V, V^c$ constructed in (4.64).

- $\alpha = 1, \beta = 0$: If $\hat{p}(1, 0, \mathbf{c}) + \hat{z}_u + \hat{z}_w < 1$, then we have found a violated inequality of type (4.33) for the $i, j, U, U^c, W, W^c, V, V^c$ constructed in (4.64).
- $\alpha = 1, \beta = 1$: If $\hat{p}(1, 1, \mathbf{c}) + \hat{z}_u + (1 - \hat{z}_w) < 1$, then we have found a violated inequality of type (4.34) for the $i, j, U, U^c, W, W^c, V, V^c$ constructed in (4.64).

3. Family 3: For a given partial solution $(\hat{u}, \hat{w}, \hat{v}, \hat{z}_u, \hat{z}_w)$, we determine if there are indices $i, j \in [r]$ and sets U, U^c, W, W^c, V, V^c such that $U \cup U^c = W \cup W^c = V \cup V^c = [r] \setminus \{i, j\}, U^c \cap W^c \cap V^c = \emptyset, U \cap V^c = \emptyset$, and the corresponding Family 3 inequality violates the partial solution $(\hat{u}, \hat{w}, \hat{v}, \hat{z}_u, \hat{z}_w)$. To separate Family 3, we solve the integer program (4.62) for parameters $(\alpha, \beta) \in \{(0, 0), (0, 1), (1, 0), (1, 1)\}$ with with $\mathbf{c} = [c_1, c_2, c_3, c_4, c_5, c_6, c_7]$ where $c_l \; \forall l \in \{1, 2, \dots 7\}$ corresponds to Family 3 in Table 4.1. The solution to integer program (4.62) can be mapped to i, j and sets of interest U, U^c, W, W^c, V, V^c as follows.

$$\begin{aligned} &i = s_1, j = s_2, U \cap W \cap V = \hat{S}_3, U \cap W^c \cap V = \hat{S}_4 \\ &U^c \cap W \cap V = \hat{S}_5, U^c \cap W^c \cap V = \hat{S}_6, U^c \cap W \cap V^c = \hat{S}_7 \end{aligned} \tag{4.65}$$

- $\alpha = 0, \beta = 0$: If $\hat{p}(0, 0, \mathbf{c}) + \hat{z}_u + 2(1 - \hat{z}_w) < 2$, then we have found a violated inequality of type (4.35) for the $i, j, U, U^c, W, W^c, V, V^c$ constructed in (4.65).
- $\alpha = 0, \beta = 1$: If $\hat{p}(0, 1, \mathbf{c}) + \hat{z}_u + 2\hat{z}_w < 2$, then we have found a violated inequality of type (4.36) for the $i, j, U, U^c, W, W^c, V, V^c$ constructed in (4.65).
- $\alpha = 1, \beta = 0$: If $\hat{p}(1, 0, \mathbf{c}) + 1 - \hat{z}_u + 2(1 - \hat{z}_w) < 2$, then we have found a violated inequality of type (4.37) for the $i, j, U, U^c, W, W^c, V, V^c$ constructed in (4.65).
- $\alpha = 1, \beta = 1$: If $\hat{p}(1, 1, \mathbf{c}) + 1 - \hat{z}_u + 2\hat{z}_w < 2$, then we have found a violated inequality of type (4.38) for the $i, j, U, U^c, W, W^c, V, V^c$ constructed in (4.65).

4. Family 4: For a given partial solution $(\hat{u}, \hat{w}, \hat{v}, \hat{z}_u, \hat{z}_w)$, we determine if there are indices $i, j \in [r]$ and sets U, U^c, W, W^c, V, V^c such that $U \cup U^c = W \cup W^c = V \cup V^c = [r] \setminus \{i, j\}, U^c \cap W^c \cap V^c = \emptyset, U \cap V^c = \emptyset$, and the corresponding Family 4 inequality violates the partial solution $(\hat{u}, \hat{w}, \hat{v}, \hat{z}_u, \hat{z}_w)$. To separate Family 4, we solve the integer program (4.62) for parameters $(\alpha, \beta) \in \{(0, 0), (0, 1), (1, 0), (1, 1)\}$ with $\mathbf{c} = [c_1, c_2, c_3, c_4, c_5, c_6, c_7]$ where $c_l \; \forall l \in \{1, 2, \dots 7\}$ corresponds to Family 4 in Table 4.1. The solution to integer program (4.62) can be mapped to i, j and sets of

interest U, U^c, W, W^c, V, V^c as follows.

$$\begin{aligned} &i = s_1, j = s_2, U \cap W \cap V = \hat{S}_3, U \cap W^c \cap V = \hat{S}_4 \\ &U^c \cap W \cap V = \hat{S}_5, U^c \cap W^c \cap V = \hat{S}_6, U^c \cap W \cap V^c = \hat{S}_7 \end{aligned} \tag{4.66}$$

- $\alpha = 0, \beta = 0$: If $\hat{p}(0, 0, \mathbf{c}) + \hat{z}_u + 2(1 - \hat{z}_w) < 2$, then we have found a violated inequality of type (4.35) for the $i, j, U, U^c, W, W^c, V, V^c$ constructed in (4.66).
- $\alpha = 0, \beta = 1$: If $\hat{p}(0, 1, \mathbf{c}) + \hat{z}_u + 2\hat{z}_w < 2$, then we have found a violated inequality of type (4.36) for the $i, j, U, U^c, W, W^c, V, V^c$ constructed in (4.66).
- $\alpha = 1, \beta = 0$: If $\hat{p}(1, 0, \mathbf{c}) + 1 - \hat{z}_u + 2(1 - \hat{z}_w) < 2$, then we have found a violated inequality of type (4.37) for the $i, j, U, U^c, W, W^c, V, V^c$ constructed in (4.66).
- $\alpha = 1, \beta = 1$: If $\hat{p}(1, 1, \mathbf{c}) + 1 - \hat{z}_u + 2\hat{z}_w < 2$, then we have found a violated inequality of type (4.38) for the $i, j, U, U^c, W, W^c, V, V^c$ constructed in (4.66).

4.4.4 Heuristic Separation

We now discuss an approach to generate a valid inequality for each $(i, j) \in [r] \times [r], i \neq j$ for a given $(\hat{u}, \hat{w}, \hat{v}, \hat{z}_u, \hat{z}_w)$ pair, i.e., we fix (i, j) and then find a violated inequality. We let $r' = r - 2$ and $[r'] = [r] \setminus \{i, j\}$. To find a violated inequality after fixing (i, j), we minimize left-hand-side for each of the valid inequalities proposed in Theorems 4.15-4.18, and observe that the minimization problem reduces to partitioning set $[r']$ in 5 different sets, say $\mathcal{S} = (S_1, S_2, \ldots, S_5)$.

Let $\mathbf{c} = [c_1, c_2, \ldots, c_5]$ where $c_j \in \mathbb{R}^{r'}\ \forall j = 1, \ldots, 5$. Let θ_j^l be a binary variable which takes a value of 1 if index $l \in [r']$ is assigned to set $S_j\ \forall j = 1, \ldots, 5$. We also let θ_j denote $[\theta_j^1, \theta_j^2, \ldots, \theta_j^{r'}]$.

$$\min\ c_1^\top \theta_1 + c_2^\top \theta_2 + c_3^\top \theta_3 + c_4^\top \theta_4 + c_5^\top \theta_5 \tag{4.67a}$$

$$\sum_{j=1}^{5} \theta_j^l = 1 \quad \forall l \in [r'] \tag{4.67b}$$

$$\theta \in \{0, 1\}^{r' \times 5} \tag{4.67c}$$

It is easy to see that above problem has a closed form solution as shown in Algorithm 6.

Algorithm 6: Subroutine for heuristic separation of valid inequalities

Input: Vectors c_1, c_2, c_3, c_4, c_5 and indices i, j

Output: Partition of $[r] \setminus \{i, j\}$ in sets S_1, S_2, S_3, S_4, S_5

$[r'] = [r] \setminus \{i, j\}$;
$S_l = \{\}\ \ \forall l \in \{1, 2, \ldots 5\}$;
for $k \in [r']$ **do**
    $l^* = \arg\min\{c_{lk}\ \forall l \in \{1, 2, \ldots, 5\}\}$;
    /* Break tie by letting l^* be the smallest index */
    $S_{l^*} \leftarrow S_{l^*} \cup \{k\}$;
end
return $\hat{\mathcal{S}} = (S_1, S_2, S_3, S_5, S_5)$

Algorithm 7: 1-swap heuristic

Input: Candidate set (S) , target set (T), cost coefficients $\mathbf{c}$

Output: S' and T'

$c_s = \mathbf{c}[S]$; /* r dimensional vector */
$c_T = \mathbf{c}[T]$; /* r dimensional vector */
$k_1 = \arg\min\{-c_S[k] + c_T[k]\ \forall k \in S\}$; /* Move element from S to T */
$c_1 = -c_S[k_1] + c_T[k_1]$;
$k_2 = \arg\min\{c_S[k] - c_T[k]\ \forall k \in T\}$; /* Move element from T to S */
$c_2 = c_S[k_2] - c_T[k_1]$;
if $c_1 \leq c_2$
    $S' \leftarrow S \setminus \{k_1\}, T' \leftarrow T \cup \{k_1\}$; /* Expanding T is cheaper */
else
    $S' \leftarrow S \cup \{k_2\}, T' \leftarrow T \setminus \{k_2\}$; /* Shrinking T is cheaper */
end
return S', T';

We further define two more subroutines in Algorithm 8 to change cardinality of sets $U \cap V$ and $W \cap V$.

We let $\alpha = 0$ if $|U \cap V|$ is even and $\alpha = 1$ if $|U \cap V|$ is odd. Similarly, we let $\beta = 0$ if $|W \cap V|$ is even and $\beta = 1$ if $W \cap V$ is odd.

1. Family 1: For all $i, j \in [r], i \neq j$, we generate a partition of $[r'] = [r] \setminus \{i, j\}$ using Algorithm 6 for $\mathbf{c} = [c_1, c_2, c_3, c_4, c_5]$ where $c_l\ \forall l \in \{1, 2, \ldots 5\}$ corresponds to Family 1

Algorithm 8: Procedure for changing cardinality of $U \cap V$ and $W \cap V$

procedure CHANGE UV($\mathbf{c}, U^c \cap W^c \cap V, V^c, U \cap W^c \cap V$)
Invoke 1-swap heuristic Alg. 7 for $S = (U^c \cap W^c \cap V) \cup V^c, T = U \cap W^c \cap V$

procedure CHANGE WV($\mathbf{c}, U^c \cap W^c \cap V, V^c, U^c \cap W \cap V$)
Invoke 1-swap heuristic Alg. 7 for $S = (U^c \cap W^c \cap V) \cup V^c, T = U^c \cap W \cap V$

procedure CHANGE UWV($\mathbf{c}, U^c \cap W^c \cap V, V^c, U^c \cap W \cap V$)
Invoke 1-swap heuristic Alg. 7 for $S = (U^c \cap W^c \cap V) \cup V^c, T = U \cap W^c \cap V$
Update current partition
Invoke 1-swap heuristic Alg. 7 for $S = (U^c \cap W^c \cap V) \cup V^c, T = U^c \cap W \cap V$

		Desired config.			
	(α, β)	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$
Current config.	$(0,0)$	-	*CHANGE WV*	*CHANGE UV*	*CHANGE UWV*
	$(0,1)$	*CHANGE WV*	-	*CHANGE UWV*	*CHANGE UV*
	$(1,0)$	*CHANGE UV*	*CHANGE UWV*	-	*CHANGE WV*
	$(1,1)$	*CHANGE UWV*	*CHANGE UV*	*CHANGE WV*	-

Table 4.2: Swapping procedure to invoke from Algorithm 8 to transition from current (α, β) configuration to desired (α, β) configuration.

	Family 1	Family 2	Family 3	Family 4
c_1	$3 - \hat{u} - \hat{w} - \hat{v}$	$2 - \hat{u} - \hat{w}$	$4 - \hat{u} - 2\hat{w} - \hat{v}$	$4 - \hat{u} - 2\hat{w} - \hat{v}$
c_2	$2 - \hat{u} - \hat{v} + \hat{w}$	$2 - \hat{u} + \hat{w} - \hat{v}$	$2 - \hat{u} + 2\hat{w} - \hat{v}$	$2 - \hat{u} + 2\hat{w} - \hat{v}$
c_3	$2 + \hat{u} - \hat{w} - \hat{v}$	$2 + \hat{u} - \hat{w} - \hat{v}$	$4 + \hat{u} - 2\hat{w} - 2\hat{v}$	$4 + \hat{u} - 2\hat{w} - 2\hat{v}$
c_4	$\hat{u} + \hat{w}$	$\hat{u} + \hat{w}$	$\hat{u} + 2\hat{w}$	$\hat{u} + 2\hat{w}$,
c_5	$\hat{v}$	$\hat{v}$	$2\hat{v}$	$2\hat{v}$

Table 4.3: Coefficient calculation for valid inequalities separation heuristic

in Table 4.3. We map the output $\hat{\mathcal{S}}$ to our sets of interest as follows:

$$U \cap W \cap V = \hat{S}_1, U \cap W^c \cap V = \hat{S}_2, U^c \cap W \cap V = \hat{S}_3, \\ U^c \cap W^c \cap V = \hat{S}_4, (U \cup W) \cap V^c = \hat{S}_5 \tag{4.68}$$

Let's first consider the case when output from Algorithm 6 has both $|U \cap V|$ and $|W \cap V|$ as even.

(i) If $c(\hat{\mathcal{S}}) + \hat{v}_i + (1 - \hat{w}_j) + 1 - \hat{z}_u + \hat{z}_w + < 1$, then we have found a violated inequality of type (4.27).

(ii) To find a violated inequality of type (4.28), we invoke *CHANGE WV* procedure from Algorithm 8 as suggested in Table 4.2. Let $\hat{\mathcal{S}}'$ be the new partition after the swap. If $c(\hat{\mathcal{S}}') + \hat{v}_i + (1 - \hat{w}_j) + 1 - \hat{z}_u + 1 - \hat{z}_w < 1$, then we have found a violated inequality of type (4.28).

(iii) To find a violated inequality of type (4.29), we invoke *CHANGE UV* procedure from Algorithm 8 as suggested in Table 4.2. Let $\hat{\mathcal{S}}'$ be the new partition after the swap. If $c(\hat{\mathcal{S}}') + \hat{v}_i + (1 - \hat{w}_j) + \hat{z}_u + \hat{z}_w < 1$, then we have found a violated inequality of type (4.29).

(iv) To find a violated inequality of type (4.30), we invoke *CHANGE UWV* procedure from Algorithm 8 as suggested in Table 4.2. Let $\hat{\mathcal{S}}'$ be the new partition after the swap. If $c(\hat{\mathcal{S}}') + \hat{v}_i + (1 - \hat{w}_j) + \hat{z}_u + 1 - \hat{z}_w < 1$, then we have found a violated inequality of type (4.30).

Similar procedure is repeated for the cases when output from Algorithm 6 has $|U \cap V|, |W \cap V| \in \{$ odd, even $\}$ by invoking the necessary *CHANGE* procedures using Table 4.2.

2. Family 2: For all $i, j \in [r], i \neq j$, we generate a partition of $[r'] = [r] \setminus \{i, j\}$ using Algorithm 6 for $\mathbf{c} = [c_1, c_2, c_3, c_4, c_5]$ where $c_l \; \forall l \in \{1, 2, \ldots 5\}$ corresponds to Family 2 in Table 4.3. We map the output $\hat{\mathcal{S}}$ to our sets of interest as follows:

$$U \cap W \cap V = \hat{S}_1, U \cap W^c \cap V = \hat{S}_2 \\ U^c \cap W \cap V = \hat{S}_3, U^c \cap W^c \cap V = \hat{S}_4, (U \triangle W) \cap V^c = \hat{S}_5 \tag{4.69}$$

Let's first consider the case when output from Algorithm 6 has both $|U \cap V|$ and $|W \cap V|$ as even.

(i) If $c(\hat{\mathcal{S}}) + (2 - \hat{u}_i - \hat{w}_i) + (2 - \hat{u}_j - \hat{w}_j) + 1 - \hat{z}_u + \hat{z}_w < 1$, then we have found a violated inequality of type (4.31).

(ii) To find a violated inequality of type (4.32), we invoke *CHANGE WV* procedure from Algorithm 8 as suggested in Table 4.2. Let $\hat{\mathcal{S}}'$ be the new partition after the swap. If $c(\hat{\mathcal{S}}') + (2 - \hat{u}_i - \hat{w}_i) + (2 - \hat{u}_j - \hat{w}_j) + 1 - \hat{z}_u + 1 - \hat{z}_w < 1$, then we have found a violated inequality of type (4.32).

(iii) To find a violated inequality of type (4.33), we invoke *CHANGE UV* procedure from Algorithm 8 as suggested in Table 4.2. Let $\hat{\mathcal{S}}'$ be the new partition after the swap. If $c(\hat{\mathcal{S}}') + (2 - \hat{u}_i - \hat{w}_i) + (2 - \hat{u}_j - \hat{w}_j) + \hat{z}_u + \hat{z}_w < 1$, then we have found a violated inequality of type (4.33).

(iv) To find a violated inequality of type (4.34), we invoke *CHANGE UWV* procedure from Algorithm 8 as suggested in Table 4.2. Let $\hat{\mathcal{S}}'$ be the new partition after the swap. If $c(\hat{\mathcal{S}}') + (2 - \hat{u}_i - \hat{w}_i) + (2 - \hat{u}_j - \hat{w}_j) + \hat{z}_u + 1 - \hat{z}_w < 1$, then we have found a violated inequality of type (4.34).

Similar procedure is repeated for the cases when output from Algorithm 6 has $|U \cap V|, |W \cap V| \in \{$ odd, even $\}$ by invoking the necessary *CHANGE* procedures using Table 4.2.

3. Family 3: For all $i, j \in [r], i \neq j$, we generate a partition of $[r'] = [r] \setminus \{i, j\}$ using Algorithm 6 for $\mathbf{c} = [c_1, c_2, c_3, c_4, c_5]$ where $c_l \; \forall l \in \{1, 2, \ldots 5\}$ corresponds to Family 3 in Table 4.3. We map the output $\hat{\mathcal{S}}$ to our sets of interest as follows:

$$\begin{aligned} &U \cap W \cap V = \hat{S}_1, U \cap W^c \cap V = \hat{S}_2 \\ &U^c \cap W \cap V = \hat{S}_3, U^c \cap W^c \cap V = \hat{S}_4, U^c \cap W \cap V^c = \hat{S}_5 \end{aligned} \tag{4.70}$$

Let's first consider the case when output from Algorithm 6 has both $|U \cap V|$ and $|W \cap V|$ as even.

(i) If $c(\hat{\mathcal{S}}) + (1 - \hat{u}_i + \hat{v}_i) + (1 - \hat{u}_j + \hat{w}_j) + \hat{z}_u + 2(1 - \hat{z}_w) < 2$, then we have found a violated inequality of type (4.35).

(ii) To find a violated inequality of type (4.36), we invoke *CHANGE WV* procedure from Algorithm 8 as suggested in Table 4.2. Let $\hat{\mathcal{S}}'$ be the new partition after the swap. If $c(\hat{\mathcal{S}}) + (1 - \hat{u}_i + \hat{v}_i) + (1 - \hat{u}_j + \hat{w}_j) + \hat{z}_u + 2\hat{z}_w < 2$, then we have found a violated inequality of type (4.36).

(iii) To find a violated inequality of type (4.37), we invoke *CHANGE UV* procedure from Algorithm 8 as suggested in Table 4.2. Let $\hat{\mathcal{S}}'$ be the new partition after

the swap. If $c(\hat{\mathcal{S}}) + (1 - \hat{u}_i + \hat{v}_i) + (1 - \hat{u}_j + \hat{w}_j) + 1 - \hat{z}_u + 2(1 - \hat{z}_w) < 2$, then we have found a violated inequality of type (4.37).

(iv) To find a violated inequality of type (4.38), we invoke *CHANGE UWV* procedure from Algorithm 8 as suggested in Table 4.2. Let $\hat{\mathcal{S}}'$ be the new partition after the swap. If $c(\hat{\mathcal{S}}) + (1 - \hat{u}_i + \hat{v}_i) + (1 - \hat{u}_j + \hat{w}_j) + 1 - \hat{z}_u + 2\hat{z}_w < 2$, then we have found a violated inequality of type (4.38).

Similar procedure is repeated for the cases when output from Algorithm 6 has $|U \cap V|, |W \cap V| \in \{$ odd, even $\}$ by invoking the necessary *CHANGE* procedures using Table 4.2.

4. Family 4: For all $i, j \in [r], i \neq j$, we generate a partition of $[r'] = [r] \setminus \{i, j\}$ using Algorithm 6 for $\mathbf{c} = [c_1, c_2, c_3, c_4, c_5]$ where $c_l \; \forall l \in \{1, 2, \ldots 5\}$ corresponds to Family 4 in Table 4.3. We map the output $\hat{\mathcal{S}}$ to our sets of interest as follows:

$$\begin{aligned} &U \cap W \cap V = \hat{S}_1, U \cap W^c \cap V = \hat{S}_1 \\ &U^c \cap W \cap V = \hat{S}_3, U^c \cap W^c \cap V = \hat{S}_4, U^c \cap W \cap V^c = \hat{S}_5 \end{aligned} \tag{4.71}$$

Let's first consider the case when output from Algorithm 6 has both $|U \cap V|$ and $|W \cap V|$ as even.

(i) If $c(\hat{\mathcal{S}}) + (1 - \hat{u}_i + \hat{v}_i) + (1 - \hat{u}_j + \hat{w}_j) + \hat{z}_u + 2(1 - \hat{z}_w) < 2$, then we have found a violated inequality of type (4.42).

(ii) To find a violated inequality of type (4.36), we invoke *CHANGE WV* procedure from Algorithm 8 as suggested in Table 4.2. Let $\hat{\mathcal{S}}'$ be the new partition after the swap. If $c(\hat{\mathcal{S}}) + (1 - \hat{u}_i + \hat{v}_i) + (1 - \hat{u}_j + \hat{w}_j) + \hat{z}_u + 2\hat{z}_w < 2$, then we have found a violated inequality of type (4.43).

(iii) To find a violated inequality of type (4.37), we invoke *CHANGE UV* procedure from Algorithm 8 as suggested in Table 4.2. Let $\hat{\mathcal{S}}'$ be the new partition after the swap. If $c(\hat{\mathcal{S}}) + (1 - \hat{u}_i + \hat{v}_i) + (1 - \hat{u}_j + \hat{w}_j) + 1 - \hat{z}_u + 2(1 - \hat{z}_w) < 2$, then we have found a violated inequality of type (4.44).

(iv) To find a violated inequality of type (4.38), we invoke *CHANGE UWV* procedure from Algorithm 8 as suggested in Table 4.2. Let $\hat{\mathcal{S}}'$ be the new partition after the swap. If $c(\hat{\mathcal{S}}) + (1 - \hat{u}_i + \hat{v}_i) + (1 - \hat{u}_j + \hat{w}_j) + 1 - \hat{z}_u + 2\hat{z}_w < 2$, then we have found a violated inequality of type (4.45).

Similar procedure is repeated for the cases when output from Algorithm 6 has $|U \cap V|, |W \cap V| \in \{$ odd, even $\}$ by invoking the necessary *CHANGE* procedures using Table 4.2.

4.5 Computational Study

In this section we compare the formulations proposed in Section 4.2. In addition, we also study the performance of valid inequalities proposed in Section 4.4.1. In particular, we compare the following formulation:

1. Base formulation: Formulation I (4.6)
2. Disjunctive formulation: Formulation II (4.9)
3. SQT formulation: Formulation III (4.13)
4. SQT formulation with valid inequalities proposed in Section 4.4.1. We consider exact separation proposed in Section 4.4.3 as well as the heuristic separation strategy proposed in Section 4.4.4. We separate the proposed inequalities only at root node in both the cases.

4.5.1 Experimental Setup

We implement the proposed formulations in Python and use Gurobi 10.0 as the integer programming solver. We set a time limit of 7200s for each formulation. The reported experiments were performed on a cluster of 6 core 16 GB machines with Xeon X5690 CPU running at 3.46GHz. We test our models on synthetically generated datasets. We generate a basis matrix U by sampling each entry U_{ik} for all $(i, k) \in [d] \times [r]$ from a Bernoulli distribution with probability $P(U_{ik} = 1) = 0.5$. Similarly, we generate each entry V_{kj} for all $(k, j) \in [r] \times [n]$ from a Bernoulli distribution with probability $P(V_{kj} = 1) = 0.5$. For a column vector V_j of V, if the resulting vector $X_j = UV_j$ is repeated, we resample all entries for the vector V_j. Thus, we do not allow identical column vectors in the data matrix X. After generating the data matrix X, we uniformly at random drop a percentage f of the entries in X. We generate matrices with $d \in \{6, 8, 10\}, n \in \{8, 10, 15\}, r \in \{4, 5, 6, 7\}, f \in \{0, 10, 20, 30\}$. We refer to r as the true rank or the underlying rank with which data matrix X was generated. We use r_f to denote the rank of decomposition model we fit. For a given rank r of the data matrix X, we fit the decomposition model with $r_f \in \{r - 2, r - 1, r\}$.

Table 4.4: Summary of the solved instances

r_f	# instances	# instances with zero error on observed entries
2	59	0
3	100	0
4	105	82

4.5.2 Metrics

We compare the proposed formulations on the following metrics:

- Number of nodes explored in branch-and-bound.
- Time taken in branch-and-bound. Here we do not include time spent in generating valid inequalities at root node.
- Solution time. Here we compare the total solution time for each of the formulations including the cut separation time.

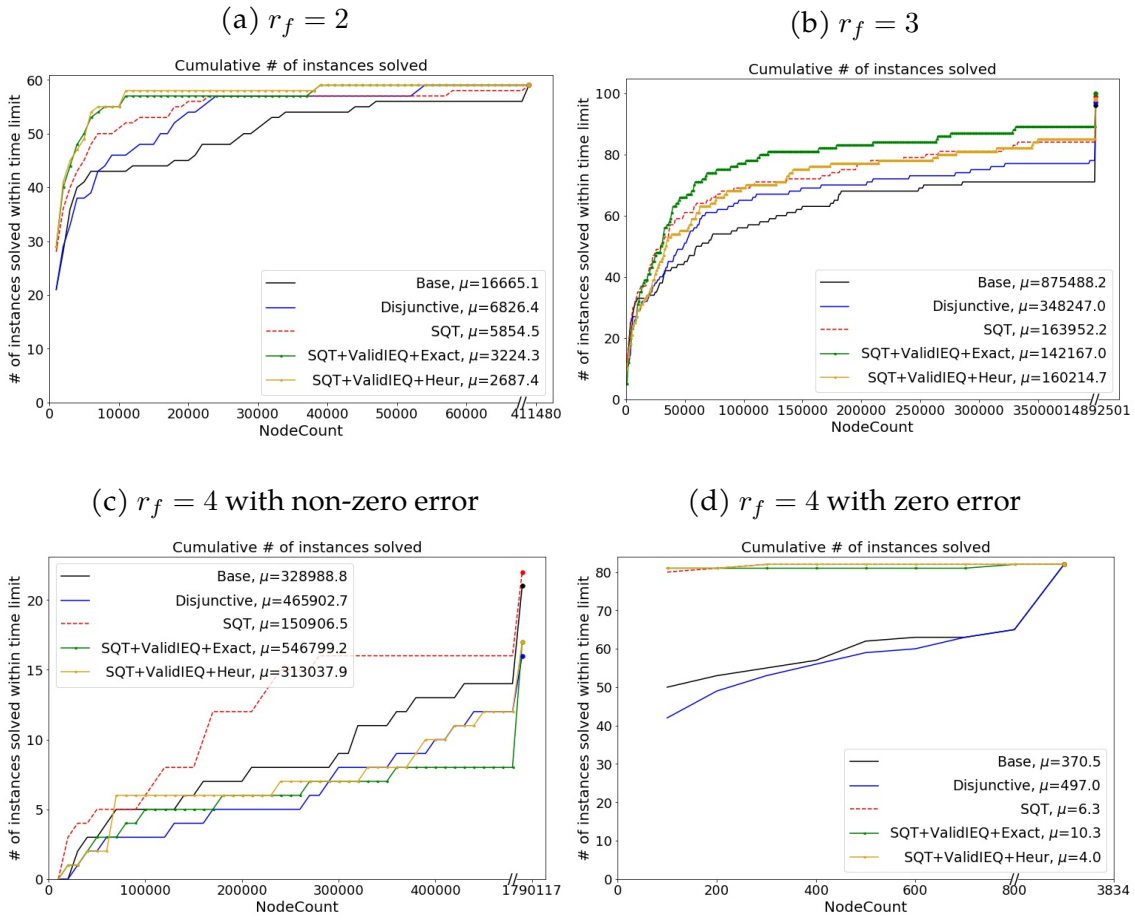
4.5.3 Comparison of Formulations

We report a summary of the number of instances solved in Table 4.4. We first note that for $r_f \in \{2, 3\}$, none of the instances resulted in zero error on observed entries. However, for $r_f = 4$, 82 of the total 105 instances resulted in 0 error on observed entries. This observation will be useful in analyzing the results we discuss next.

We compare the number of nodes explored in branch-and-bound algorithm for each of the formulations. We plot the cumulative number of instances solved (y-axis) against the number of nodes explored in branch-and-bound (x-axis) as shown in Figure 4.1. The arithmetic mean of number of nodes explored for each method (μ) is calculated based on the instances which were solved to optimality by all the methods in the given time limit. We also report the number of cuts added at root node as well as the time spent in cut generation and branch-and-bound algorithm in Table 4.5.

We first point out that heuristic cut separation strategy can potentially find more cuts than the exact separation strategy. This is because for a given pair of two rows from matrix $\hat{U}$ and one column from matrix $\hat{V}$ (or vice-versa), with exact cut separation strategy, we solve an integer program to find the most violated cut without iterating over all $(i, j) \subseteq [r] \times [r], i \neq j$. However, with heuristic cut separation strategy, we iterate over all $((i, j))$, thus potentially adding more cuts than the exact strategy.

Figure 4.1: Node count comparison for different formulations, μ denotes the arithmetic mean of instances solved to optimality within the time limit.



For $r_f = 2$, as shown in Figure 4.1a, we observe that the SQT formulation takes fewer nodes than Base as well as Disjunctive formulation by a factor 2.84 and 1.16 respectively. Adding valid inequalities at the root node further decreases the node count for SQT formulation by factor of 1.82 with an exact separation strategy and by a factor of 2.18 with a heuristic separation strategy. Thus, separating valid inequalities at root node resulted in smaller branch-and-bound trees than the SQT formulation. We observe marginally better performance with heuristic cut separations strategy compared to exact strategy. The average number of cuts added at root node for $r_f = 2$, across all the 59 instances is 397.80 with exact strategy and 519.27 with heuristic strategy, as reported in Table 4.5, We also observe that time spent in branch-and-bound is relatively small (< 10s) for all the instances with SQT formulation performing the best with an average time of 2.36s. There is non-trivial amount of time spent in cut separation since we haven't done a sophisticated implementation for cut separation strategy for either of the methods.

Table 4.5: Average computational times and cut generation summary of instances solved to optimality within the time limit.

r_f	Method	# cuts added	Time in finding cuts (s)	B&B time (s)	Nodes
		Non-zero error on observed entries			
2	Base	-	-	9.78	16665.05
2	Disjunctive	-	-	3.89	6826.39
2	SQT	-	-	2.36	5854.46
2	SQT+ValidIEQ+Exact	397.80	18.77	5.30	3224.34
2	SQT+ValidIEQ+Heur	519.27	7.92	4.70	2687.36
3	Base	-	-	316.41	875488.23
3	Disjunctive	-	-	373.38	348246.99
3	SQT	-	-	104.13	163952.23
3	SQT+ValidIEQ+Exact	925.17	500.05	104.12	142166.99
3	SQT+ValidIEQ+Heur	6819.97	502.19	223.26	160214.66
4	Base	-	-	297.49	328988.81
4	Disjunctive	-	-	1756.68	465902.69
4	SQT	-	-	303.75	150906.50
4	SQT+ValidIEQ+Exact	1386.88	1097.94	1367.96	546799.19
4	SQT+ValidIEQ+Heur	3149.25	799.52	1037.64	313037.88
		Zero error on observed entries			
4	Base	-	-	0.59	370.54
4	Disjunctive	-	-	5.96	497.02
4	SQT	-	-	0.30	6.30
4	SQT+ValidIEQ+Exact	4151.66	2092.49	0.46	10.30
4	SQT+ValidIEQ+Heur	8879.82	1690.41	0.53	3.98

For $r_f = 3$, we observe similar performance of the Base and Disjunctive formulations when compared to the SQT formulation. Cut separation marginally reduced the number of nodes explored but is not as effective as in the case of $r_f = 2$. In particular, the exact strategy reduced number of nodes by 1.13 and heuristic strategy by 1.02 when compared to the SQT formulation. We also highlight that on an average, only 925.17 cuts were added by the exact cut separation strategy and adding cuts did not affect the time spent in branch-and-bound compared to the SQT formulation. On the other hand, the heuristic strategy added 6819.97 cuts and yet performed worse than the exact strategy in nodes explored. Moreover, adding more cuts also resulted in increasing the time spent in branch-and-bound by a factor of 2. In this case, cuts added with the exact strategy were more effective than the cuts generated by heuristic strategy, as one might expect.

For $r_f = 4$, we consider two separate cases based on whether the error on observed entries is non-zero (23 of the considered 105 instances), or the error is zero (82 of the considered 105 instances). We first point out that there is a high-variance in the number

of nodes explored for the non-zero error case compared to the zero-error case. This can be attributed to the fact that the objective function has a trivial lower bound of 0. When Gurobi's primal heuristics discover a feasible solution with 0 objective, the branch-and-bound process terminates. Notably, for instances where the error on observed entries is zero, we observed that Gurobi's primal heuristics were highly efficient in quickly identifying such solutions.

We also observe that for the non-zero error case, the SQT formulation is no longer faster than Base formulation. The number of constraints in SQT formulation are $\mathcal{O}(3^r)$, increasing exponentially with r, and hence the linear programs solved during branch-and-bound require longer computational times.

We next observe that for both the cases, adding cuts did not result in smaller branch-and-bound trees. We anticipate that the cuts added for $r_f \in \{2, 3\}$ proved to more effective than $r_f = 4$ because the valid inequalities were originally derived for a rank 2 system which were then lifted. We next observe from Figure 4.1d and Table 4.5 that when error on observed entries is 0, all formulations resulted in significantly smaller branch-and-bound trees than previous cases. The SQT based formulations have an average of < 10 nodes while Base and Disjunctive formulations have averages of 370.54 and 497.02 nodes respectively. Moreover, all formulations except the Disjunctive formulation took < 1s on an average. As one can observe from Table 4.5, a significant amount of time is still spent on finding valid inequalities at root node.

Although, the new proposed inequalities in Section 4.4.1 can reduce number of nodes in some cases, the additional new cuts do not improve solution times of SQT formulation on these test instances. The SQT formulation outperforms the disjunctive formulation in terms of the number of nodes explored and solution times. However, as r_f increases, owing to the large number of inequalities in SQT formulation ($\mathcal{O}(3^{r_f})$), the performance of the SQT formulation does not exceed that of the base formulation in terms of solution times.

4.6 Conclusion and future directions

In this chapter, we proposed three different formulations for matrix factorization and completion in $\mathbb{F}_2$. We first derived two McCormick based formulations where we first used a general integer variable to model parity sum and then used ideas of parity polytopes and disjunctive programming to model the parity sum. We then proposed a novel class of SQT inequalities which characterized the convex hull of the dot product in $\mathbb{F}_2$ and provided a formulation in the original space of variables. A disadvantage of the SQT formulation is that we need $\mathcal{O}(3^r)$ inequalities for a valid formulation and hence it might not scale well

with r. We thus discussed a linear-time separations strategy for the inequalities. We then derived new classes of inequalities linking two rows of U and discussed an exact as well as heuristic separation strategy.

One possible future direction for this work is to model parity set $\mathcal{P} = \{(y, z) \in \{0,1\}^{r+1} \mid z = \bigoplus_{k\in[r]} y\}$ by introducing variables $t_k \; \forall k \in [r]$ and imposing constraints that $t_k \geq t_{k+1}$ and $1^\top t = 1^\top y$. Variable z can then be modeled as

$$z = \sum_{k:k \text{ is odd}} t_k - \sum_{k:k \text{ is even}} t_k.$$

Another interesting direction to explore would be to see if there is a compact extended formulation for $\mathrm{conv}(\mathcal{I})$ which is linear in number of variables and constraints, and does not make use of disjunction of the parity polytopes. Since it exists for parity polytopes (switching formulation in [67]), it would be interesting to see if there is a similar formulation for convex hull of dot product in $\mathbb{F}_2$.

5 CONCLUSIONS AND FUTURE DIRECTIONS

In this thesis, we developed mixed-integer programming based methods for problems in scheduling and matrix completion.

In Chapter 2, we studied the problem of scheduling drilling and fracturing of wells in the development of an unconventional oil field. We presented a novel MILP-based rolling horizon algorithm to schedule drilling and fracturing operations in an unconventional oil field development while considering the interaction effects between various pads. We provided two MILP formulations for the limited horizon MILP solved as part of this approach. The second formulation provides better LP relaxation bounds, which translates to shorter solution times for instances with a small number of time periods, but for larger instances the first formulation was found to be more effective. A key feature of the rolling horizon approach we propose is that it yields a solution at the daily time-scale, while solving a sequence of coarser time-scale MILP problems. An empirical study demonstrated that the approach can be used to plan development of fields with more than 100 pads, and the solutions obtained have 4-6% higher NPV than solutions obtained with a baseline scheduling algorithm that mimics current practice.

Our work assumed all data, including the drilling and fracturing durations, are deterministic. In reality, these are estimated via forecasts that may have significant errors. The rolling horizon framework we propose can naturally be applied in this setting, by using the updated state of the system, and updated estimates of the durations, whenever a new limited-horizon MILP is solved. However, an interesting direction for future work is to investigate the use of a stochastic or robust optimization formulation of the scheduling problem within the rolling horizon framework to see if this may yield improved solutions.

In Chapter 3, we studied an integer programming approach for subspace clustering with missing data. *Subspace clustering with missing data* (SCMD) is the task of identifying clusters of vectors belonging to the same subspace in a partially observed data matrix whose columns are assumed to lie in a union of K subspaces. We proposed a novel mixed-integer linear programming (MILP) solution framework for this 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. The proposed framework readily accommodates 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.

MISS-DSG is computationally more expensive than the other clustering algorithms but a parallel implementation can offer significant speedups. A possible future direction is to generalize this framework to a union of models (e.g, clustering mixture of Gaussian models) instead of limiting to a union of linear subspaces.

In Chapter 4, we studied matrix factorization and completion over $\mathbb{F}_2$ and proposed three different integer programming formulations. We first derived two McCormick based formulations where we first used a general integer variable to model parity sum and then used ideas of parity polytopes and disjunctive programming to model the parity sum. We then proposed a novel class of SQT inequalities which characterized the convex hull of the dot product in $\mathbb{F}_2$ and provided a formulation in the original space of variables. A disadvantage of the SQT formulation is that we need $\mathcal{O}(3^r)$ inequalities for a valid formulation and hence it might not scale well with r. We thus discussed a linear-time separations strategy for the inequalities. We then derived new classes of inequalities linking two rows of U and discussed an exact as well as heuristic separation strategy.

One possible future direction for this work is to model parity set $\mathcal{P} = \{(y, z) \in \{0,1\}^{r+1} \mid z = \bigoplus_{k\in[r]} y\}$ by introducing variables $t_k \; \forall k \in [r]$ and imposing constraints that $t_k \geq t_{k+1}$ and $1^\top t = 1^\top y$. Variable z can then be modeled as

$$z = \sum_{k:k \text{ is odd}} t_k - \sum_{k:k \text{ is even}} t_k.$$

Another interesting direction to explore would be to see if there is a compact extended formulation for $\mathrm{conv}(\mathcal{I})$ which is linear in number of variables and constraints, and does not make use of disjunction of the parity polytopes. Another fascinating direction of research would be to advance methods for subspace clustering over $\mathbb{F}_2$ by building upon the algorithms proposed in Chapters 3 and 4.

REFERENCES

[1] Abdolali, Maryam, and Nicolas Gillis. 2021. Beyond linear subspace clustering: A comparative study of nonlinear manifold clustering algorithms. *Computer Science Review* 42:100435.

[2] Arbabjolfaei, Fatemeh, and Young-Han Kim. 2018. Fundamentals of Index Coding. *Foundations and Trends® in Communications and Information Theory* 14(3-4):163–346.

[3] Arps, J. J. 1945. Analysis of decline curves. *Transactions of the AIME* 160(1).

[4] Arredondo-Ramirez, K., J.M. Ponce-Ortega, and M.M. El-Halwagi. 2016. Optimal planning and infrastructure development for shale gas production. *Energy Conversion and Management* 119:91–100.

[5] Bah, Bubacarr, and Jannis Kurtz. 2020. An integer programming approach to deep neural networks with binary activation functions. *arXiv preprint arXiv:2007.03326*. .

[6] Balas, Egon. 1998. Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics* 89(1):3–44.

[7] Balzano, L., R. Nowak, and B. Recht. 2010. Online identification and tracking of subspaces from highly incomplete information. In *2010 48th annual allerton conference on communication, control, and computing (allerton)*, 704–711.

[8] Balzano, L., A. Szlam, B. Recht, and R. Nowak. 2012. K-subspaces with missing data. In *2012 ieee statistical signal processing workshop (ssp)*, 612–615.

[9] Balzano, Laura Kathryn. 2012. Handling missing data in high-dimensional subspace modeling. Ph.D. thesis, The University of Wisconsin - Madison, United States – Wisconsin.

[10] Ban, Frank, Vijay Bhattiprolu, Karl Bringmann, Pavel Kolev, Euiwoong Lee, and David P. Woodruff. 2019. A PTAS for ?p-Low Rank Approximation. In *Proceedings of the 2019 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 747–766. Proceedings, Society for Industrial and Applied Mathematics.

[11] Bar-Yossef, Ziv, Yitzhak Birk, T. S. Jayram, and Tomer Kol. 2011. Index Coding With Side Information. *IEEE Transactions on Information Theory* 57(3):1479–1494.

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

[13] Bartis, James T., Tom LaTourrette, Lloyd Dixon, D. J. Peterson, and Gary Cecchine. 2004. Gauging the Prospects of a U.S. Oil Shale Industry. Tech. Rep., RAND Corporation.

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

[15] Bertsimas, Dimitris, and Michael Lingzhi Li. 2018. Interpretable matrix completion: A discrete optimization approach. *arXiv preprint arXiv:1812.06647*. .

[16] Birk, Y., and T. Kol. 1998. Informed-source coding-on-demand (ISCOD) over broadcast channels. In *Proceedings. IEEE INFOCOM '98, the Conference on Computer Communications. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Gateway to the 21st Century (Cat. No.98*, vol. 3, 1257–1264 vol.3.

[17] Bixby, Robert. 2014. Business and Mathematics: A Saga of 25 Years of Progress in Optimization.

[18] Blasiak, Anna, Robert Kleinberg, and Eyal Lubetzky. 2011. Index coding via linear programming. *arXiv preprint arXiv:1004.1379*. .

[19] Bradley, P. S., and O. L. Mangasarian. 2000. k-plane clustering. *Journal of Global Optimization* 16(1):23–32.

[20] Cacciola, Matteo, Antonio Frangioni, Xinlin Li, and Andrea Lodi. 2022. Deep Neural Networks pruning via the Structured Perspective Regularization. *arXiv preprint arXiv:2206.14056*. .

[21] Cafaro, C.D., M.G. Drouven, and Grossmann Ignacio E. 2016. Optimization models for planning shale gas well refracture treatments. *AIChE Journal* 62:4297–4307.

[22] Cafaro, C.D., and I.E. Grossmann. 2014. Strategic planning, design, and development of the shale gassupply chain network. *AIChE Journal* 60:2122–2142.

[23] Cafaro, D. C., M.G. Drouven, and I.E. Grossmann. 2018. Continuous-time formulations for the optimal planning of multiple refracture treatments in a shale gas well. *AIChE Journal* 64:1511–1517.

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

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

[26] Candès, Emmanuel J., and Benjamin Recht. 2009. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics* 9(6):717–772.

[27] Carvalho, M. C. A., and J.M. Pinto. 2006. A bilevel decomposition technique for the optimal planning of offshore platforms. *Brazilian Journal of Chemical Engineeringl* 23(1):67–82.

[28] Chang, Allison An. 2012. Integer optimization methods for machine learning. Thesis, Massachusetts Institute of Technology.

[29] Charles, Z., A. Jalali, and R. Willett. 2018. Sparse subspace clustering with missing and corrupted data. In *2018 ieee data science workshop (dsw)*, 180–184.

[30] Conforti, M., G. Cornuejols, and G. Zambelli. 2014. *Integer programming*. Springer Publishing Company, Incorporated.

[31] Conforti, Michele, Marco Di Summa, and Yuri Faenza. 2017. Balas formulation for the union of polytopes is optimal. .

[32] Crafton, J., and S. Noe. 2013. Factors affecting early well productivity in six shale plays. *SPE Annual Technical Conference and Exhibition,Society of Petroleum Engineers*.

[33] Dan, Chen, Kristoffer Arnsfelt Hansen, He Jiang, Liwei Wang, and Yuchen Zhou. 2017. Low Rank Approximation of Binary Matrices: Column Subset Selection and Generalizations. .

[34] DeSantis, Derek, Erik Skau, Duc P. Truong, and Boian Alexandrov. 2021. Factorization of Binary Matrices: Rank Relations, Uniqueness and Model Selection of Boolean Decomposition. .

[35] Drouven, M., and I.E. Grossmann. 2017. Stochastic programming models for optimal shale well development and refracturing planning under uncertainty. *AIChE Journal* 63(11):4799–4813.

[36] Drouven, M. G., and I.E. Grossmann. 2016. Multi-period planning, design, and strategic models for long-term, quality-sensitive shale gas development. *Process Systems Engineeringl* 62:2296–2323.

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

[38] Elhamifar, Ehsan. 2016. High-rank matrix completion and clustering under self-expressive models. In *Advances in neural information processing systems*, ed. D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, vol. 29. Curran Associates, Inc.

[39] Elhamifar, Ehsan, and Rene Vidal. 2009. Sparse subspace clustering. In *2009 ieee conference on computer vision and pattern recognition*, 2790–2797.

[40] Esfahanizadeh, Homa, Farshad Lahouti, and Babak Hassibi. 2014. A matrix completion approach to linear index coding problem. In *2014 IEEE Information Theory Workshop (ITW 2014)*, 531–535.

[41] Etherington, J.R, and I. R. McDonald. 2004. Is bitumen a petroleum reserve? *SPE annual technical conference and exhibition. Society of Petroleum Engineers*.

[42] ExxonMobil. 2017. ExxonMobil to acquire companies doubling Permian Basin resource to 6 billion barrels. https://corporate.exxonmobil.com:443/news/news-releases/2017/0117_exxonmobil-to-acquire-companies-doubling-permian-basin-resource-to-6-billion-barrels.

[43] ———. 2019. ExxonMobil to increase, accelerate Permian output to 1 million barrels per day by 2024. https://corporate.exxonmobil.com:443/news/news-releases/2019/0305_exxonmobil-to-increase-accelerate-permian-output-to-1-million-barrels-per-day-by-2024.

[44] Fan, Jicong, and Tommy W.S. Chow. 2017. Matrix completion by least-square, low-rank, and sparse self-representations. *Pattern Recognition* 71:290–305.

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

[46] Fomin, Fedor, Fahad Panolan, Anurag Patil, and Adil Tanveer. 2022. Boolean and $\mathbb{F}_p$-matrix factorization: From theory to practice.

[47] Fomin, Fedor V., Petr A. Golovach, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. 2018. Approximation Schemes for Low-Rank Binary Matrix Approximation Problems. .

[48] Ford, L. R., and D. R. Fulkerson. 1958. A suggested computation for maximal multi-commodity network flows. *Management Science* 5(1):97–101. .

[49] Garey, M.R. 1979. Computers and intractability: A guide to the theory of NP-completeness. *Freeman, NewYork*.

[50] Goel, V., and I.E. Grossmann. 2004. A stochastic programming approach to planning of offshore gas field developments under uncertainty in reserves. *Computers and chemical engineering* 28(8):1409–1429.

[51] Golub, Gene H., and Charles F. Van Loan. 1996. Matrix computations.

[52] Gupta, J.N. 1988. Two-stage, hybrid flowshop scheduling problem. *Journal of Operational Research Society* 39(4):359–364.

[53] Gurobi Optimization. 2022. Gurobi 10.0. https://www.gurobi.com/whats-new-gurobi-10-0/.

[54] Harvey, Nicholas J. A., David R. Karger, and Sergey Yekhanin. 2006. The complexity of matrix completion. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm - SODA '06*, 1103–1111. Miami, Florida: ACM Press.

[55] Hong, Wei, John Wright, Kun Huang, and Yi Ma. 2006. Multiscale Hybrid Linear Models for Lossy Image Representation. *IEEE Transactions on Image Processing* 15(12): 3655–3671.

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

[57] Huang, Kun, Yi Ma, and R. Vidal. 2004. Minimum effective dimension for mixtures of subspaces: a robust gpca algorithm and its applications. In *Proceedings of the 2004 ieee computer society conference on computer vision and pattern recognition, 2004. cvpr 2004.*, vol. 2, II–II.

[58] Huang, Xiao, and Salim El Rouayheb. 2015. Index coding and network coding via rank minimization. In *2015 IEEE Information Theory Workshop - Fall (ITW)*, 14–18. Jeju Island, South Korea: IEEE.

[59] Iyer, R. R., I. E. Grossmann, S. Vasantharajan, and A. S. Cullick. 1998. Optimal planning and scheduling of offshore oil field infrastructure investment and operations. *Industrial and Engineering Chemistry Researchl* 37:1380–1397.

[60] Klotz, Ed, and Alexandra M. Newman. 2013. Practical guidelines for solving difficult mixed integer linear programs. *Surveys in Operations Research and Management Science* 18(1):18–32.

[61] Knudsen, B.R., and B. Foss. 2013. Shut-in based production optimization of shale-gas systems. *Computers and Chemical Engineering* 58:54–67.

[62] Koch, Thorsten, Timo Berthold, Jaap Pedersen, and Charlie Vanaret. 2022. Progress in mathematical programming solvers from 2001 to 2020. *EURO Journal on Computational Optimization* 10:100031.

[63] Kondili, E., C.C. Pantelidas, and R.W.H. Sargent. 1993. A general algorithm for short-term scheduling of batch operations, milp formulation. *Computers and Chemical Engineering* 17:211–227.

[64] Koster, Arie M. C. A., Manuel Kutschka, and Christian Raack. 2010. Towards robust network design using integer linear programming techniques. In *6th EURO-NGI Conference on Next Generation Internet*, 1–8.

[65] Kovacs, Reka A., Oktay Gunluk, and Raphael A. Hauser. 2021. Binary Matrix Factorisation via Column Generation. *Proceedings of the AAAI Conference on Artificial Intelligence* 35(5):3823–3831.

[66] Kumar, Ravi, Rina Panigrahy, Ali Rahimi, and David Woodruff. 2019. Faster Algorithms for Binary Matrix Factorization. In *Proceedings of the 36th International Conference on Machine Learning*, 3551–3559. PMLR.

[67] Lancia, Giuseppe, and Paolo Serafini. 2018. *Compact Extended Linear Programming Models*. EURO Advanced Tutorials on Operational Research, Cham: Springer International Publishing.

[68] Lane, Connor, Ron Boger, Chong You, Manolis Tsakiris, Benjamin Haeffele, and Rene Vidal. 2019. Classifying and comparing approaches to subspace clustering with missing data. In *Proceedings of the ieee/cvf international conference on computer vision (iccv) workshops*.

[69] Lawless, Connor, and Oktay Günlük. 2020. Fair and interpretable decision rules for binary classification. In *NeurIPS Workshop*.

[70] Lazic, Nevena, Inmar Givoni, Brendan Frey, and Parham Aarabi. 2009. Floss: Facility location for subspace segmentation. In *2009 ieee 12th international conference on computer vision*, 825–832.

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

[72] Lee, C., and L. Cheong. 2013. Minimal basis facility location for subspace segmentation. In *2013 ieee international conference on computer vision (iccv)*, 1585–1592. Los Alamitos, CA, USA: IEEE Computer Society.

[73] Lee, Daniel D., and H. Sebastian Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401(6755):788–791.

[74] Lee, T.S., and Y.T. Loong. 2019. A review of scheduling problem and resolution methods in flexible flow shop. *International Journal of Industrial Engineering Computations* 10:67–88.

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

[76] Li, Tao. 2005. A general model for clustering binary data. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 188–197.

[77] Liberti, Leo, Sonia Cafieri, and Fabien Tarissan. 2009. Reformulations in Mathematical Programming: A Computational Approach. In *Foundations of Computational Intelligence Volume 3: Global Optimization*, ed. Ajith Abraham, Aboul-Ella Hassanien, Patrick Siarry, and Andries Engelbrecht, 153–234. Studies in Computational Intelligence, Berlin, Heidelberg: Springer.

[78] Lin, X., and A.F. Christodoulos. 2003. A novel continuous-time modeling and optimization framework for well platform planning problems. *Optimization and Engineering* 4:65–95.

[79] Liu, Guangcan, Zhouchen Lin, and Yong Yu. 2010. Robust subspace segmentation by low-rank representation. In *Proceedings of the 27th international conference on international conference on machine learning*, 663–670. ICML'10, Madison, WI, USA: Omnipress.

[80] Lu, Can-Yi, Hai Min, Zhong-Qiu Zhao, Lin Zhu, De-Shuang Huang, and Shuicheng Yan. 2012. Robust and efficient subspace segmentation via least squares regression. In *Computer vision – eccv 2012*, ed. Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, 347–360. Berlin, Heidelberg: Springer Berlin Heidelberg.

[81] Lu, Canyi, Jiashi Feng, Zhouchen Lin, and Shuicheng Yan. 2013. Correlation adaptive subspace segmentation by trace lasso. In *2013 ieee international conference on computer vision*, 1345–1352.

[82] Manchanda, R., P. Bhardwaj, J. Hwang, and M. Sharma. 2018. Parent-child fracture interference: Explanation and mitigation of child well underperformance. *SPE Annual Technical Conference and Exhibition,Society of Petroleum Engineers*.

[83] Marquant, J. F., R. Evins, and J. Carmeliet. 2015. Reducing computation time with a rolling horizon approach applied to a milp formulation of multiple urban energy hub system. *ICCS 2015 International Conference On Computational Science* 2137–2146.

[84] McCormick, Garth P. 1976. Computability of global solutions to factorable nonconvex programs: Part I — Convex underestimating problems. *Mathematical Programming* 10(1):147–175.

[85] Miettinen, Pauli, and Stefan Neumann. 2021. Recent developments in Boolean matrix factorization. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 4922–4928. IJCAI'20, Yokohama, Yokohama, Japan.

[86] Morais, Hugo, Péter Kádár, Pedro Faria, Zita A. Vale, and H. M. Khodr. 2010. Optimal scheduling of a renewable micro-grid in an isolated load area using mixed-integer linear programming. *Renewable Energy* 35(1):151–156.

[87] Natarajan, Nagarajan, and Inderjit Dhillon. 2014. Inductive matrix completion for predicting gene-disease associations. *Bioinformatics (Oxford, England)* 30:i60–i68.

[88] Ng, Andrew Y., Michael I. Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, 849–856. MIT Press.

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

[90] Office of Fossil Energy. 2013. Natural gas from shale. *U.S. Department of Energy*.

[91] Ondeck, A., M. Drouven, N. Blandino, and I.E. Grossmann. 2019. Multi-operational planning of shale gas pad development. *Computers and Chemical Engineering* 126: 83–101.

[92] Panagakis, Yannis, and Constantine Kotropoulos. 2014. Elastic net subspace clustering applied to pop/rock music structure analysis. *Pattern Recognition Letters* 38: 46–53.

[93] Pimentel, D., R. Nowak, and L. Balzano. 2014. On the sample complexity of subspace clustering with missing data. In *2014 ieee workshop on statistical signal processing (ssp)*, 280–283.

[94] Pimentel-Alarcón, Daniel L., and R. Nowak. 2016. The information-theoretic requirements of subspace clustering with missing data. In *Icml*.

[95] Pimentel-Alarcón, D., L. Balzano, R. Marcia, R. Nowak, and R. Willett. 2016. Group-sparse subspace clustering with missing data. In *2016 ieee statistical signal processing workshop (ssp)*, 1–5.

[96] Pimentel-Alarcón, Daniel L., Nigel Boston, and Robert D. Nowak. 2015. A characterization of deterministic sampling patterns for low-rank matrix completion. In *2015 53rd annual allerton conference on communication, control, and computing (allerton)*, 1075–1082.

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

[98] Pourhejazy, Pourya, and Oh Kyoung Kwon. 2016. The New Generation of Operations Research Methods in Supply Chain Optimization: A Review. *Sustainability* 8(10): 1033.

[99] Rahmanifard, H., and T. Plaksina. 2018. Application of fast analytical approach and ai optimization techniques to hydraulic fracture stage placement in shale gas reservoirs. *Journal of Natural Gas Science and Engineering* 52:367–378.

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

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

[102] Recht, Benjamin. 2011. A simpler approach to matrix completion. *J. Mach. Learn. Res.* 12(null):3413–3430.

[103] Rudin, Cynthia, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. 2022. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys* 16:1–85.

[104] Ryan, David M., and Brian A. Foster. 1981. An integer programming approach to scheduling. *Computer scheduling of public transport urban passenger vehicle and crew scheduling* 269–280.

[105] Samà, M., A. D'Ariano, and D. Pacciarelli. 2013. Rolling horizon approach for aircraft scheduling in the terminal control area of busy airports. *Procedia-Social and Behavioral Sciences* 80:531–552.

[106] Saunderson, James, Maryam Fazel, and Babak Hassibi. 2016. Simple algorithms and guarantees for low rank matrix completion over F2. In *2016 IEEE International Symposium on Information Theory (ISIT)*, 86–90.

[107] Silvente, J., G. M. Kopanos, E. N. Pistikopoulos, and A. Espuna. 2015. A rolling horizon optimization framework for the simultaneous energy supply and demand planning in microgrids. *Applied Energy* 155:485–501.

[108] Soltanolkotabi, Mahdi, and Emmanuel J. Candés. 2012. A geometric analysis of subspace clustering with outliers. *The Annals of Statistics* 40(4):2195–2238.

[109] Spratt, B., and E. Kozan. 2018. An integrated rolling horizon approach to increase operating theatre efficiency. .

[110] Tan, Vincent Y. F., Laura Balzano, and Stark C. Draper. 2012. Rank Minimization Over Finite Fields: Fundamental Limits and Coding-Theoretic Interpretations. *IEEE Transactions on Information Theory* 58(4):2018–2039.

[111] Tarhan, B., I.E. Grossmann, and V. Goel. 2009. Stochastic programming approach for the planning of offshore oil or gas field infrastructure under decision-dependent uncertainty. *Industrial and Engineering Chemistry Research* 48(6):3078–3097.

[112] Tron, Roberto, and Rene Vidal. 2007. A benchmark for the comparison of 3-d motion segmentation algorithms. In *2007 ieee conference on computer vision and pattern recognition*, 1–8.

[113] Tsakiris, Manolis, and Rene Vidal. 2018. Theoretical analysis of sparse subspace clustering with missing entries. In *Proceedings of the 35th international conference on machine learning*, ed. Jennifer Dy and Andreas Krause, vol. 80 of *Proceedings of Machine Learning Research*, 4975–4984. Stockholmsmässan, Stockholm Sweden: PMLR.

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

[115] U.S. Energy Information Administration EIA . 2019. Tight oil development will continue to drive future U.S. crude oil production. *Today in Energy*.

[116] Van Loan, Charles F., and G. Golub. 1996. Matrix computations (Johns Hopkins studies in mathematical sciences). *Matrix Computations*.

[117] Vidal, R., S. Soatto, Yi Ma, and S. Sastry. 2003. An algebraic geometric approach to the identification of a class of linear hybrid systems. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, vol. 1, 167–172 Vol.1.

[118] Wang, H. 2016. Numerical investigation of fracture spacing and sequencing effects on multiple hydraulic fracture interference and coalescence in brittle and ductile reservoir rocks. *Engineering Fracture Mechanics* 157:107–124.

[119] Wang, Yu-Xiang, Huan Xu, and Chenlei Leng. 2019. Provable subspace clustering: When lrr meets ssc. *IEEE Transactions on Information Theory* 65(9):5406–5432.

[120] Wicker, Jörg, Yan Cathy Hua, Rayner Rebello, and Bernhard Pfahringer. 2019. XOR-Based Boolean Matrix Decomposition. In *2019 IEEE International Conference on Data Mining (ICDM)*, 638–647.

[121] Xie, J., and X. Wang. 2005. Complexity and algorithms for two-stage flexible flowshop scheduling with availability constraints. *Computers and Mathematics with Applications* 50(10-12):1629–1638.

[122] Yang, Congyuan, Daniel Robinson, and Rene Vidal. 2015. Sparse subspace clustering with missing entries. In *Proceedings of the 32nd international conference on machine learning*, ed. Francis Bach and David Blei, vol. 37 of *Proceedings of Machine Learning Research*, 2463–2472. Lille, France: PMLR.

[123] You, Chong, Chun-Guang Li, Daniel P. Robinson, and René Vidal. 2016. Oracle based active set algorithm for scalable elastic net subspace clustering. In *2016 ieee conference on computer vision and pattern recognition (cvpr)*, 3928–3937.

[124] Zapata, E. L., J. Gonzalez-Mora, F. De la Torre, N. Guil, and R. Murthi. 2007. Bilinear active appearance models. In *2007 11th ieee international conference on computer vision*, 1–8. Los Alamitos, CA, USA: IEEE Computer Society.

[125] Zhuang, Liansheng, Haoyuan Gao, Zhouchen Lin, Yi Ma, Xin Zhang, and Nenghai Yu. 2012. Non-negative low rank and sparse graph for semi-supervised learning. In *2012 ieee conference on computer vision and pattern recognition*, 2328–2335.