SPARSITY CONSTRAINED MINIMAX OPTIMIZATION WITH APPLICATIONS TO GAME THEORY AND MACHINE LEARNING

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF

MASTER OF SCIENCE

IN

INDUSTRIAL ENGINEERING

By

Bora Çetin

July 2025

SPARSITY CONSTRAINED MINIMAX OPTIMIZATION WITH APPLICATIONS TO GAME THEORY AND MACHINE LEARNING

By Bora Çetin

July 2025

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Mustafa Çelebi Pınar(Advisor)

Firdevs Ulus

Cem İyigün

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

SPARSITY CONSTRAINED MINIMAX OPTIMIZATION WITH APPLICATIONS TO GAME THEORY AND MACHINE LEARNING

Bora Çetin
M.S. in Industrial Engineering
Advisor: Mustafa Çelebi Pınar
July 2025

Classical game-theoretic methods often yield dense strategies for a player, which might not be practical in real-world implementations. This thesis studies incorporating the sparsity constraint to the minimax optimization problem to compute sparse strategies in bimatrix games. The theory and algorithms also apply to the equivalent problem of computing sparse classifiers in the margin maximizing boosting problem.

Optimality conditions in the sparse optimization literature are extended to nonsmooth functions. A new optimality condition for neighborhood search that covers the existing conditions is proposed. Practical greedy algorithms are developed to find candidate points satisfying optimality conditions.

Based on the properties of the Minimax function, connections between the cardinality-constrained problem and the cardinality-regularized problem are established. A new concave penalty for cardinality-regularized optimization problems over the unit simplex is proposed, which offers an alternative to the sparsity-promoting penalties in the literature. The resulting problem is solved efficiently using a faster version of the Difference of Convex (DC) algorithm.

The proposed algorithms are tested empirically on random game matrices and real data for binary classification. The performance is compared to well-known regularization techniques and the MILP formulation of the problem.

Keywords: Minimax, Sparsity, Regularization, DCA.

ÖZET

OYUN TEORİSİ VE MAKİNE ÖĞRENİMİ UYGULAMALARIYLA SEYREKLİK KISITLI MİNİMUM-MAKSİMUM OPTİMİZASYON

Bora Çetin
Endüstri Mühendisliği, Yüksek Lisans
Tez Danışmanı: Mustafa Çelebi Pınar
Temmuz 2025

Klasik oyun teorisi yöntemleri bir oyuncu için çoğunlukla yoğun stratejiler üretmektedir; ancak bu durum, gerçek dünya uygulamalarında pratik olmayabilir. Bu tez, iki oyunculu oyunlarda seyrek stratejileri hesaplamak için minimum-maksimum eniyileme problemine seyreklik kısıtını dahil etmeyi incelemektedir. Teori ve algoritmalar, bu probleme eşdeğer olan sınırı maksimize eden artırma probleminde seyrek sınıflandırıcıların hesaplanmasında da uygulanabilmektedir.

Öncelikle, seyrek optimizasyon literatüründeki en iyilik koşulları türevi olmayan fonksiyonlar için genişletildi. Ayrıca, halihazırda bulunan bu koşulları kapsayan ve komşu araması yapmayı sağlayan yeni bir koşul geliştirildi. Bu koşulları sağlayan en iyiliğe aday noktaları bulma amacıyla pratik açgözlü algoritmalar geliştirildi.

Minimum-maksimum fonksiyonunun özellikleri kullanılarak, seyreklik kısıtlı problem ve seyreklik düzenleyicili problemin bağlantıları oluşturuldu. Literatürdeki seyreklik artırıcı cezalara bir alternatif olarak, birim simpleks üzerinde seyreklik-düzenleyicili eniyileme problemleri için yeni bir içbükey ceza önerildi. Elde edilen problem Dışbükey Farkı (DC) Algoritması'nın hızlandırılmış bir versiyonu ile çözülebilmektedir.

Önerilen algoritmalar oyun teorisi için rastgele oluşturulmuş matrisler ve ikili sınıflandırma için gerçek veriler üzerinde deneysel olarak test edildi. Algoritmaların performansı, iyi bilinen düzenleme teknikleri ve problemin MILP formülasyonuyla karşılaştırıldı.

Anahtar sözcükler: Minimum-maksimum, Seyreklik, Düzenleme, DCA.

Acknowledgement

I would like to share my profound gratitude for those who supported me throughout my Master's studies.

Firstly, I wish to thank my thesis advisor, Mustafa Ç. Pınar for his support from day one to the completion of this thesis. Many things would not have been possible without his guidance, wisdom, and patience.

I am grateful to my professors in Bilkent: Firdevs Ulus, Çağın Ararat, Ülkü Gürler, M. Selim Aktürk, and Bahar Yetiş, for everything they have taught me, including but not up to theory, conducting top quality research, and how a great teacher should be.

I would also like to share my sincere gratitude to my thesis committee members, Firdevs Ulus and Cem İyigün, for their helpful recommendations.

I wish to thank my family: my mother, Sebahat Çetin, my father, Peyami Çetin, and my brother, Ata Çetin, for their unconditional love and trust in me.

Completing this Master's thesis would not be possible without the warmest support of my lifelong friends: Turan Yılmaz, Kutup Aytekin, Emre Kasapoğlu, Ali Çağan Namak, Arca Alan, Arda Arslanyürek and Ali Suveren; and my friends from Bilkent: Yiğit Ateş, Ali Bingöl, Öykü İman, Özgün Barış Kır, Can Baran Kocatürk, Zeynep Nalbant, and Deniz Yayla.

Furthermore, I wish to send my sincerest gratitude to all my friends in the graduate school who made this journey of my Master's bearable: Kaan Çakıroğlu, Sena Aslı Bozkurt, Efecan Şentürk, Doğuş Berk Koçak, Osman Kağan Yayla, Muhittin Can Korkut, and Deniz Akkaya.

Finally, I also wish to thank TÜBİTAK for their financial support during my Master's Degree studies through the 2210-A National MSc/MA Scholarship Programs.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis aims to find solutions to the Minimax optimization problem under cardinality (sparsity) and the unit simplex constraints. In general, a Minimax problem is an optimization problem of the form

$$\min_{\mathbf{y}\in\mathcal{Y}} \max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x},\mathbf{y}), \tag{1.1}$$

where $\mathcal{X}$ and $\mathcal{Y}$ are any form of finite dimensional spaces and $f(\mathbf{x},\mathbf{y}) : \mathcal{X}\times\mathcal{Y} \to \mathbb{R}$ is the bilinear utility function as in [1]. Minimax problem finds applications in various areas including Game Theory, Graph Theory and Computational Optimization [2].

A well-known family of examples from the Game Theory are the Zero-Sum games. Solving (1.1) allows computing the saddle point of a zero sum game with $f(\mathbf{x},\mathbf{y}) = \mathbf{x}^\intercal\mathbf{A}\mathbf{y}$, $\mathcal{Y} = \Delta^n$, $\mathcal{X} = \Delta^m$, where $\mathbf{A} \in \mathbb{R}^{m\times n}$ is the payoff matrix associated with the zero-sum game and Δ^k denotes the $k \in \mathbb{N}$ dimensional unit simplex defined as

$$\Delta^k := \{x \in \mathbb{R}^k \mid 1^\intercal\mathbf{x} = 1, \mathbf{x} \succeq 0\}. \tag{1.2}$$

Minimax problems have been extensively studied since the famous *Minimax Theorem* of von Neumann [3], which states that for a zero sum game the following equality holds

$$\min_{\mathbf{y}\in\Delta^n} \max_{\mathbf{x}\in\Delta^m} \mathbf{x}^\intercal A\mathbf{y} = \max_{\mathbf{x}\in\Delta^m} \min_{\mathbf{y}\in\Delta^n} \mathbf{x}^\intercal A\mathbf{y}.$$

In this context, given a game matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, the column player finds optimum strategies by solving the optimization problem

$$\begin{aligned} \text{minimize} \quad & \max \mathbf{a}_i^T \mathbf{x} \\ \text{subject to} \quad & \mathbf{x} \in \Delta^n. \end{aligned} \tag{1.3}$$

where $\mathbf{a}_i \in \mathbb{R}^n$ corresponds to rows of matrix $\mathbf{A}$. Solving (1.3) is computationally easy, as it is a convex optimization problem. However, strategies obtained by (1.3) might be *dense*, i.e., it might contain a large number of nonzero variables, especially for large n. Dense solutions are efficient in theory. On the other hand, they might be practically infeasible and very hard to implement, and are usually criticized under the *Bounded Rationality* phenomenon in Economics. [4] Under this criticism, sub-optimal but simple solutions might be desirable in practical applications. [5].

An intuitive way of handling simplicity in strategies is to introduce cardinality constraints of the form

$$\|\mathbf{x}\|_0 \leq k, \tag{1.4}$$

where $\mathbf{x} \in \mathbb{R}^n$ is any decision variable, $k \in \mathbb{N}$ is a cardinality limit to the zero pseudo-norm, defined as

$$\|\mathbf{x}\|_0 := \sum_{i=1}^{n} \mathbf{1}_{\{\mathbf{x}_i \neq 0\}}(\mathbf{x}). \tag{1.5}$$

Incorporating this constraint provides a strict limit on the nonzero elements of a variable. Any vector $\mathbf{x} \in \mathbb{R}^n$ satisfying (1.4) is called a k-sparse vector. Throughout this thesis, the terms "sparsity" and "cardinality" are used interchangeably.

This thesis aims to analyze the optimization problem (1.3) under the cardinality constraints (1.4) by constructing optimality and stationarity conditions, developing theoretically supported heuristic algorithms, and establishing connections between similar problems in the literature. The problem of this thesis is called *Cardinality Constrained Minimax Optimization Problem* (CCMOP). A summary of proposed methods can be found in Figure 1.

The stationarity and necessary optimality conditions are constructed based on the existing work on the simpler versions of cardinality-constrained optimization

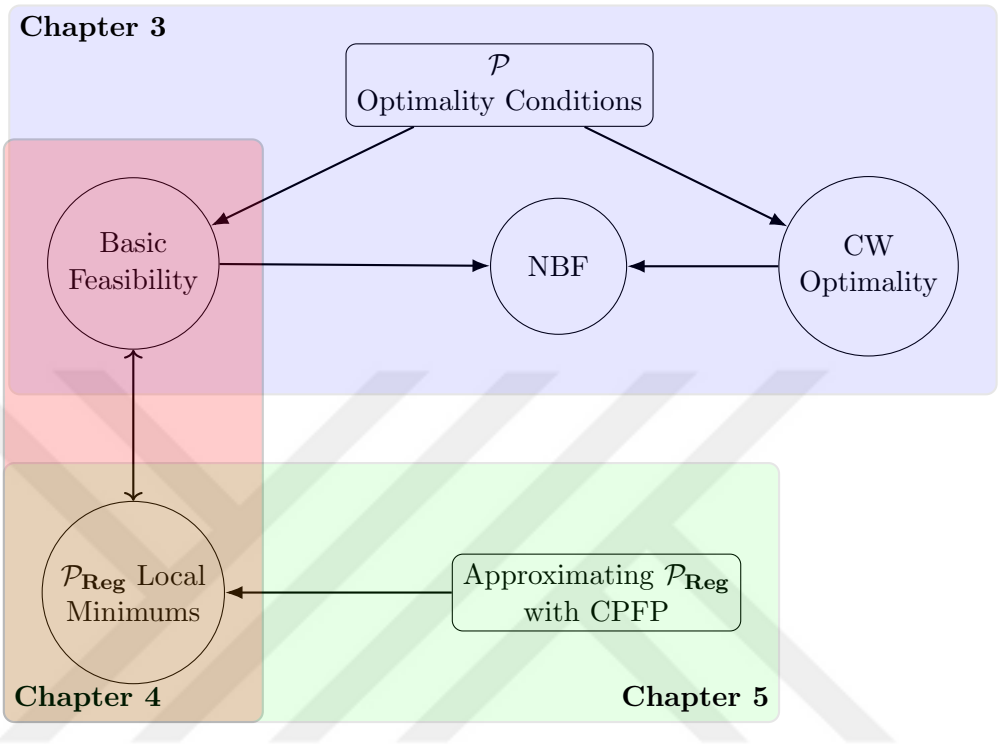

Figure 1.1: Layout of the Methods

problems. The initial attempts to solve the CCMOP-like problems include greedy algorithms to find points satisfying the necessary optimality conditions. As an alternative to the constrained problem, the Cardinality Regularized Optimization Problems (CROP) class is introduced and solved by developing a novel continuous penalty function for Difference of Convex (DC) programming. As a special case of CROP, one can formulate the *Cardinality Regularized Minimax Optimization Problem* (CRMOP).

For practical reasons, the DC algorithm is modified for the matrix games, and it has been shown that it has a column that promotes/demotes interpretation in Minimax optimization. Using theoretical results, the DCA is simplified significantly. The performance of the Greedy-like Algorithm and the Modified DC Algorithm are tested versus the Mixed-integer approaches in the literature.

1.1 Problem Definition and Properties

Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ be the game matrix of a two player game, and $\mathbf{a}_i \in \mathbb{R}^n$ denote its rows. Moreover, the decision variable $\mathbf{x} \in \mathbb{R}^n$ denotes the strategies of the column player in this game. The CCMOP can be formulated as follows.

$$\begin{array}{ll} \text{minimize} & \max \mathbf{a}_i^T \mathbf{x} \\ \text{subject to} & \mathbf{x} \in \Delta^n \\ & \|\mathbf{x}\|_0 \leq k. \end{array} \tag{$\mathcal{P}$}$$

The unit simplex Δ^n and the zero norm $\|\mathbf{x}\|_0$ are as in (1.2) and (1.4), respectively, and $k \in \mathbb{N}$ is the cardinality limit. Note that this problem is meaningful as long as $k < n$, otherwise $\mathcal{P}$ reduces to (1.3) and can be solved efficiently.

Problem $\mathcal{P}$ falls into the category of cardinality (sparsity) constrained optimization problems. As it was shown in Chapter 2, $\mathcal{P}$ has several distinctive properties: Firstly, the objective function $\max \mathbf{a}_i^T \mathbf{x}$ is convex but non-smooth, yet by convexity its directional derivatives exist. Secondly, the domain is non-convex due to (1.4), which poses extra challenges in optimization.

Vast majority of the problems involving constraint (1.4), are NP-Hard in general and hence cannot be solved efficiently, unless P=NP [6]. Indeed, an attempt of finding k-sparse strategies in $\mathcal{P}$ can be done by setting $n - k$ elements of $\mathbf{x}$ to zero. Each of those subproblems are similar to (1.3) in nature and can be solved efficiently. However, there are $C(n, k)$ many such problems, and thus, finding the exact solution is still significantly expensive.

1.2 Applications in Binary Classification

Although the main motivation of this thesis is from the perspective of computing sparse strategies in Game Theory, the problem $\mathcal{P}$ finds an application in boosting sparse classifiers in binary classification. In Machine Learning, sparsity has been studied under sparse regression and sparse support vector machines (SVM) [7].

Boosting algorithms are machine learning methods that first appeared in 1990's [8]. The task is to train a classifier $H : \mathcal{X} \to \{-1, 1\}$ which is *good enough* to predict a binary outcome from any point in $\mathcal{X}$. In this setting, $d \in \mathbb{N}$ many data points are collected in the data set D. Each data point is classified under n-many classifiers, $h_j : \mathcal{X} \to \{-1, 1\}$ from the set of weak classifiers, denoted by $\mathcal{H}$. For each data point x_i, the correct label is denoted by $y_i \in \{-1, 1\}$ for $i \in \{1, \ldots, m\}$ to train and test this data. The aim in boosting is to combine each $h_i, i \in \{1, \ldots, n\}$, with a nonnegative weight to obtain a composite classifier h, defined as

$$h = \sum_{i=1}^{n} \alpha_i h_i,$$

where $\mathbf{1}^\intercal \alpha = 1$ and $\alpha \succeq 0$. For any $x \in \mathcal{X}$ one can then assign binary predictions according to the procedure

$$H(x) = \begin{cases} 1 & \text{if } \operatorname{sgn}(h(x)) \geq 0 \\ -1 & \text{if } \operatorname{sgn}(h(x)) < 0. \end{cases}$$

The interest is in solutions with the *margin-maximizing* property, meaning that one seeks $\alpha \in \Delta$ such that

$$\alpha^* \in \arg\max_{\alpha \in \Delta} \min_{i \in D} \sum_{j=1}^{n} \alpha_j y_i h_j(x_i).$$

This interpretation combined with (1.4) forms the Margin Maximizing Sparse Composite Classifier Problem (MMSCCP), formulated as

$$\begin{aligned} \text{maximize} \quad & \min_{i \in D} \sum_{j=1}^{n} \alpha_j y_i h_j(x_i) \\ \text{subject to} \quad & \alpha \in \Delta^n \\ & \|\alpha\|_0 \leq k. \end{aligned} \tag{1.6}$$

Note that when the objective coefficients are replaced with

$$z_{ij} = -y_i h_j(x_i),$$

for every $i \in D$ and $j \in \{1, \ldots, n\}$, the problem becomes identical to $\mathcal{P}$. The matrix Z will be called the *error matrix* of a given problem parameters with $(\mathrm{x}, \mathrm{y}, \mathrm{h})$. Henceforth, solving $\mathcal{P}$ is equivalent to solving the MMSCCP. This interpretation is quite useful in real world applications, as binary classification

is a well-studied topic, and experimental data can be found available online in many open source platforms. The applications of binary classification vary from healthcare, image retrieval and object detection [9]. In conclusion, the theory and algorithms developed in the subsequent chapters are amenable to the boosting context. Therefore, the methodology is applied to examples from both Game Theory and Binary Classification.

Chapter 2

Literature Review

Despite the fact that $\mathcal{P}$ is a simple formulation, it lacks most of the common assumptions in the literature, such as differentiability and hence Lipschitz continuity of its gradient. Therefore, if possible, most literature involves finding approximate solutions to $\mathcal{P}$-like problems with a theoretical guarantee. Another common approach in the literature is studying the regularized problem and its approximations. A comprehensive survey of similar problems is provided in [10]. However, most of the examples discussed in this survey are from signal processing, and such examples benefit from differentiable objective functions. This thesis does not focus on the cardinality minimization problem but only the regularized problem and the constrained problem $\mathcal{P}$.

In the spirit of [10], the literature survey classifies common approaches in two main sections: Ones dealing with the cardinality-constrained problem directly and ones approximating the cardinality-constrained problems with a regularizer (penalty) in the objective.

Throughout this chapter, problems of the form

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{X} \\ & \|\mathbf{x}\|_0 \leq k. \end{aligned} \tag{CCOP}$$

are called cardinality constrained optimization problems (CCOP), whereas problems of the form

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) + \beta \|\mathbf{x}\|_0 \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{X} \end{aligned} \tag{CROP}$$

denote the family of Cardinality Regularized Optimization Problems (CROP). The literature survey covers various solution attempts to problems involving different selections of the objective $f(\mathbf{x})$ and the feasible set $\mathcal{X}$. In both problems, the unconstrained problem refers to choosing $\mathcal{X} = \mathbb{R}^n$.

2.1 Solving the Cardinality Constrained Problem

Solving cardinality-constrained problems is broadly approached in two ways: Firstly, by developing exact methods, mainly through Mixed Integer Linear Programs, to find globally optimal solutions, and secondly, by developing heuristic algorithms and constructing optimality conditions to obtain approximate solutions. This section reviews both approaches for various CCOP and CROP problems in the literature.

2.1.1 Exact Methods via Integer Programming

All CCOPs can be formulated as a mixed-integer program (MIP). A serious advantage to the exact methods through MIP formulation is that it provably yields the optimum k-sparse solution upon termination. Moreover, using the epigraph formulation of the max function, $\mathcal{P}$ can be reformulated as a mixed integer linear program (MILP). Therefore, a natural first attempt to solve $\mathcal{P}$ is to formulate it

as the following MILP.

$$
\begin{aligned}
\text{minimize} \quad & t \\
\text{subject to} \quad & \mathbf{x} \in \Delta \\
& \mathbf{a}_i^T \mathbf{x} \leq t \quad \forall i \in \{1, \ldots, m\} \\
& \mathbf{y} \succeq \mathbf{x} \\
& \sum_{i=1}^{n} y_i \leq k \\
& \mathbf{y} \in \mathbb{B}^n \\
& t \in \mathbb{R}
\end{aligned}
\tag{$\mathcal{P}_{\mathbf{MILP}}$}
$$

This is commonly known as the big-M formulation (See a similar reformulation in [11], Chapter 2), which uses auxiliary variables $\mathbf{y} \in \mathbb{B}^n$ to indicate if an element $i \in \{1, 2, \ldots, n\}$ is strictly positive or not. In this case of the big-M formulation, one can choose the variable lower and upper bounds as 0 and 1 due to the simplex constraint, respectively. This reformulation is quite similar to the *Vertex P-Center Problem* as in [12], except the fact that the number of centers is fixed and predetermined. When the number of centers P (which corresponds to k in $\mathcal{P}$) varies, the problem is a variant of $\mathcal{P}_{\mathbf{MILP}}$ and is NP-complete (see [ND51] in [13], List of NP-Complete Problems). An alternative to this formulation is the complementarity type constraints discussed in [14]. In Chapter 6, the big-M formulation is the primary benchmark of algorithms developed in this thesis, mostly because of the fact that the bounds of variable $\mathbf{x}$ are predetermined due to the unit simplex, and its relaxation is a linear program, unlike nonlinear complementarity-type constraints.

Solving CCOP via the MILP formulation, through state-of-the-art solvers, might be a reliable way of attempting to solve $\mathcal{P}$. On the other hand, the formulations are known to give poor optimality gaps and particularly loose lower bounds. [10] The experiments in Chapter 6 shows that for large instances the optimality gap is very large even after running the solver for hours. Thus, a viable strategy might be to identify stationary points of this problem and try to construct polynomial time algorithms to find those stationary points.

2.1.2 The Study of Stationary Conditions and Algorithms

The systematic study of CCOP traces back to [15] with a branch and bound algorithm modified for a cardinality-constrained quadratic optimization (CCQO) problem. By modifying the branching scheme, algorithms, and results for CCQO were later developed in [16]. These examples are explicitly tailored for CCQO. The general study of CCOP with smooth objectives first appears in [17]. In this work, the authors define three different optimality conditions: Basic Feasibility, $L-$Stationarity, and Coordinatewise Optimality, which was later extended to the constrained case under several symmetric sets in [18]. The authors propose greedy-type algorithms to find points satisfying these conditions, although no performance comparison with state-of-the-art methods was provided. Other papers studying optimality and stationarity conditions with a smooth objective function are [19], [20], [21] and [22]. Tailored algorithms for sparse optimization with unit simplex were also developed in [23] and [24]. However, these techniques also require f to be differentiable in CCOP. To the best of the author's knowledge, there is no current work on the stationary points of nonsmooth problems like $\mathcal{P}$.

2.2 Solving the Cardinality Regularized Problem

Due to computational challenges, problems of the CROP type are usually analyzed instead of the original CCOPs. One can expect that the CCOP and CROP should be related theoretically. The theoretical relation between CCOP and CROP was established in [25], [26] and [27], specifically for the minimization problem with $f(\mathbf{x}) = \|A\mathbf{x} - b\|_2^2$ and $\mathcal{X} = \mathbb{R}^n$. In [25] and [26], the author connects CCOP and CROP through the local optima of the CROP and subproblems of CCOP, which corresponds to the Basic Feasibility notion defined in [18]. Although the proofs in those papers are problem-specific, the theoretical results are expected to hold for any convex function. The family of CROPs is

discontinuous and nonconvex by nature. It is important to note that minimization involving zero norm terms is known to be an NP-hard task [28]. Therefore, solving CROP's globally is also computationally expensive. Most applications replace the discontinuity of the zero norm with a continuous penalty function to address the discontinuity of the zero norm.

2.2.1 Continuous Approximations to CROPs

Most of the approximations in the literature are of the form

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) + \beta g(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{X} \end{aligned} \tag{2.1}$$

where $g : \mathbb{R}^n \to \mathbb{R}$ is a continuous (and often concave) function that promotes sparsity in a similar fashion to $\|\mathbf{x}\|_0$, in particular, when $g(\cdot)$ is concave, it has been shown that solving 2.1 is also mostly NP-Hard [29]. Common surrogates used for sparse regularization are L_1-norm (LASSO) [30], Capped L_1(as in [31]), Minmax Concave Penalty (MCP)[32], Concave L_p-norm with $p \in (0, 1)$ [33], SCAD [34], and more recently CEL0 [35] and Piecewise Quadratic [36]. A comparative study of some of the above penalties for the $L_2 - L_0$ case can be found in [37]. Most of the formulations involving such penalties are non-convex, with the exception of the L_1 penalty, which makes it quite amenable in most large-scale applications. Hence, one possible approach to solve $\mathcal{P}$ is to use the L_1 regularization. However, the simplex constraint implies that for every $\mathbf{x} \in \Delta$, it holds that

$$\mathbf{1}^\intercal \mathbf{x}, \mathbf{x} \succeq 0 \implies \|\mathbf{x}\|_1 = 1.$$

Therefore, L_1 regularization is not effective in promoting sparsity in $\mathcal{P}$. To the best of current knowledge, there are only two papers on penalty functions designed to promote sparsity over the unit simplex. These are [38], which proposes $\frac{1}{\|\mathbf{x}\|_\infty}$ and [39], which proposes $-\|\mathbf{x}\|_2^2$, coinciding with the penalty proposed in [36] over the unit simplex. In fact, most of the penalties discussed above fail in promoting sparsity in the unit simplex, as recovering the desired sparsity level k is often impossible, or requires tuning penalty parameter β carefully. A review of

the performance and accuracy of the aforementioned surrogates, as well as other common issues, is provided in Chapter 5. This thesis aims to fill this gap in the literature by introducing a novel penalty tailored to optimization over the unit simplex.

Even with a continuous replacement of the zero norm, the resulting problem is most of the time nonconvex. This makes obtaining or verifying a global optimum very difficult. As a result, algorithms were developed to solve the problem for a promising local minimum.

2.2.2 Common Solution Methods

The solution methods were extensively developed for the CROP with $f(\mathbf{x}) = \|A\mathbf{x} - b\|_2^2$ and $\mathcal{X} = \mathbb{R}^n$. Some well-studied methods for this case are the IHT and its variants [40], Matching Pursuit [41], and Proximal Methods [42]. For a large family of CROP approximations with convex objectives and concave penalties, the Difference of Convex Algorithm (DCA) has also found some applications [43]. This algorithm is also known as the Convex-Concave Procedure [44]. A study of DC-type algorithms has been evaluated in [43] for most of the well-known penalty functions. As a recent DCA application example, in [45], the authors propose a DCA-based algorithm to solve a relaxation of the group sparsity regularized problem for general objective functions. The authors do not assume differentiability or convexity of f. However, the relaxation of the group sparsity regularized problem boils down to the L_1 regularization when the group sparsity term is discarded ($\lambda_2 = 0$).

Using DCA is advantageous, because the objective function f of CROP is convex, meaning that the subproblems of the DCA are convex problems. Considering the regularized version of $\mathcal{P}$, the subproblems are linear programs, making them even easier to compute. Therefore, this thesis focuses on solving the regularized problem via the DC algorithm.

2.3 Approaches in Game Theory and Machine Learning

The efforts to solve $\mathcal{P}$ for sparse solutions are motivated based on two applications: Finding sparse solutions for two-player games and constructing sparse composite classifiers in the boosting problem.

2.3.1 Sparse Strategies in Two-Player Games

Finding sparse strategies in games has attracted interest and can be traced back to finding *approximate* sparse strategies in games. As an interesting result in this context, [46] presents an approximation lemma by showing that for any strategy, there is an ϵ-approximating strategy with a bounded number of strictly positive elements. Similar work on the existence of ϵ-approximate simple strategies is further developed in [5] and [4]. Finding exact k-sparse solutions is not well-studied, until recently in [47], the authors propose a MILP approach to solve for k-sparse commitments in two-player games and develop fast and relatively accurate algorithms.

This thesis uses a different approach than [47], and instead of formulating integer programs, the focus is solely on the structural properties of the Minimax function. On the other hand, the aim is to form a bridge between various results from other sparse optimization contexts, such as compressed sensing, as well as feature selection in machine learning, to compute sparse strategies.

2.3.2 Sparse Classifiers in Boosting

Boosting in machine learning is a method of combining weak classifiers to obtain strong classifiers [48]. The idea in boosting is that a combination of classifiers can yield better classifiers than individual classifiers. One of the most famous boosting

algorithms is arguably AdaBoost by [8], and a recent outlook of AdaBoost can be found in [49]. The generalization performance of AdaBoost is mostly attributed to minimum margin maximization [50], although it is not guaranteed that AdaBoost finds a margin maximizing solution [51]. In [52], the authors develop LPBoost algorithm to solve the margin maximization problem via linear programming. Similar to the case with two-player games, the composite classifiers generated by LPBoost most of the time require selecting a large number of classifiers. Sparsity is not only useful for simplicity but also a good way to identify which k classifiers are the most important to maximize the minimum margin. Thus, it is beneficial to consider the sparse LPBoost problem by adding constraint 1.4 to the original formulation of [52]. Note that solving this problem is equivalent to solving $\mathcal{P}$, as per the construction in Section 1.2.

In machine learning literature, sparsity promotion is handled primarily by adding penalties or modifying the original algorithms. [53] develops a variant of AdaBoost by modifying the exponential loss minimization problem with an L_1 penalty. In [54], the authors propose an L_2 based variant of boosting to promote sparsity, called SparseL_2Boost. Furthermore, early stopping [55] in boosting is also a technique to avoid overfitting and promote sparsity. Selecting a $k-$subset of classifiers in sparse optimization is closely related to feature selection in machine learning. To promote feature selection, gradient-based methods were proposed ([56],[57]) in addition to well-known regularization methods such as LASSO [30]. For binary classification, [7] proposes an outer approximation algorithm to solve the sparse classification problem exactly.

Exact methods via approximations are out of the scope of this thesis. On the other hand, the numerical experiments in Boosting are rooted in machine learning applications.

Chapter 3

Development of The Optimality Conditions

Optimality conditions in optimization are ways of proving, disproving or checking optimality of a feasible solution. Necessary optimality conditions are conditions that must be satisfied by every global optimum point. On the other hand, sufficient optimality conditions directly imply global optimality. As an example, the optimality condition for unconstrained optimization of a differentiable function f is that every local optimum point x^* satisfies [58]

$$\nabla f(\mathbf{x}^*) = 0.$$

Most optimality conditions are easy to verify for optimization of well-behaved functions (e.g., continuous, convex) over well-behaved domains such as convex sets. Nonetheless, optimization of such problems is desired in applications. $\mathcal{P}$, on the other hand, is a nonsmooth optimization problem over a nonconvex set. Solving $\mathcal{P}$, therefore, requires problem-specific classification of its feasible points.

This chapter develops three main optimality conditions for problem $\mathcal{P}$. The notions of Basic Feasibility and Coordinatewise optimality can be seen as extensions of [18], whereas Neighbor Basic Feasibility is a new condition developed by combining those extended conditions.

3.1 Notation and Preliminaries

Let $k < n$ be a natural number denoting the sparsity limit of $\mathbf{x}$. Moreover, the index set is defined as

$$I := \{1, 2, 3, \ldots, n\},$$

consisting of indices for variable $\mathbf{x}$. Moreover set

$$M := \{1, 2, 3, \ldots, m\}.$$

For any point $\mathbf{x} \in C$, its support is denoted by

$$I_1(x) := \{i \in I \mid x_i > 0\},$$

and similarly it is possible to define the elements with zero value as

$$I_0(x) := \{i \in I \mid x_i = 0\}.$$

Throughout this chapter, assume that any subset S of I is ordered with the partial order $\leq$. For any subset $S \subseteq I$, let $\mathbf{x}^S \in \mathbb{R}^{|S|}$ denote the restriction of vector $\mathbf{x} \in \mathbb{R}^n$ to support S. That is,

$$\mathbf{x}_i^S = \mathbf{x}_{S_i},$$

where S_i denotes the ith element of set S. Moreover, for any function f of n variables, its restriction to $S \subseteq I$ is defined as as follows

$$f^S(\mathbf{x}^S) := f(\mathbf{x}), \mathbf{x}_i = 0, \forall i \in I \setminus S.$$

Let $C \subseteq \mathbb{R}^n$. The restriction of set C to indices S is given by

$$C^S := \{\mathbf{x} \in C \mid \mathbf{x}_i = 0, i \in I \setminus S\}.$$

When $S = \emptyset$, the set is simply denoted by C^n. Moreover, if the dimensions of a set's elements are obvious from the context, the dimension is not specified and the set is simply written as C. As an example, most of the equations in this work use Δ, which stands for the n-dimensional unit simplex for the case of $\mathcal{P}$. For simplicity, denote the k-sparse solutions of a set C as follows.

$$C_k := \{\mathbf{x} \in C \mid \|\mathbf{x}\|_0 \leq k\}.$$

For problem $\mathcal{P}$, we denote its optimum value by $\mathcal{P}^*$. Most of the algorithms in this chapter require computing the sparsity-relaxed version of $\mathcal{P}$. The relaxed problem is defined as follows.

$$\begin{aligned} &\text{minimize} && \max \mathbf{a}_i^T \mathbf{x} \\ &\text{subject to} && \mathbf{x} \in \Delta \end{aligned} \tag{$\mathcal{P}_{\mathbf{Rel}}$}$$

The notion of Basic Feasibility defined in the next section is introduced for the general problem with

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) \\ &\text{subject to} && \mathbf{x} \in C_k, \end{aligned} \tag{3.1}$$

where f is a convex and a continuous function on C and C is a compact and convex subset of $\mathbb{R}^n$. Throughout this thesis, the classical definitions of subgradients and subdfifferentials are used.

Definition 1 (Subgradient and Subdifferentials [59])**.** *Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be a proper convex function. The subdifferential of f at a point $x \in \mathrm{dom}(f)$ is*

$$\partial f(x) = \{g \in \mathbb{R}^n \mid f(y) \geq f(x) + \langle g, y - x \rangle \ \ \forall y \in \mathbb{R}^n\}. \tag{3.2}$$

Any vector vector $g \in \partial f(x)$ is called a subgradient of f at x.

Unless it is specified otherwise, all open balls $B(x, r)$ are L_2-norm balls centered at point x with radius r.

3.2 Basic Feasibility

Even though the domain of $\mathcal{P}$ is nonconvex, it possesses some useful properties. For example, consider the three-dimensional 2-Sparse unit simplex $\Delta_2^3 \subseteq \mathbb{R}^3$, which is visualized in Figure 3.1. One can observe that the sparse simplex can be represented as a union of convex subsets of Δ. In the above example, one can represent the sparse unit simplex by the union

$$\begin{aligned} \Delta_2^3 = &\{x \in \mathbb{R}^3 \mid x_1 + x_3 = 1, x_2 = 0, x_1, x_3 \geq 0\} \\ &\cup \{x \in \mathbb{R}^3 \mid x_2 + x_3 = 1, x_1 = 0, x_2, x_3 \geq 0\} \\ &\cup \{x \in \mathbb{R}^3 \mid x_1 + x_2 = 1, x_3 = 0, x_1, x_2 \geq 0\}. \end{aligned}$$

Each of these union terms are lower dimensional unit-simplexes, and optimization over these subsets would be convex optimization problems.

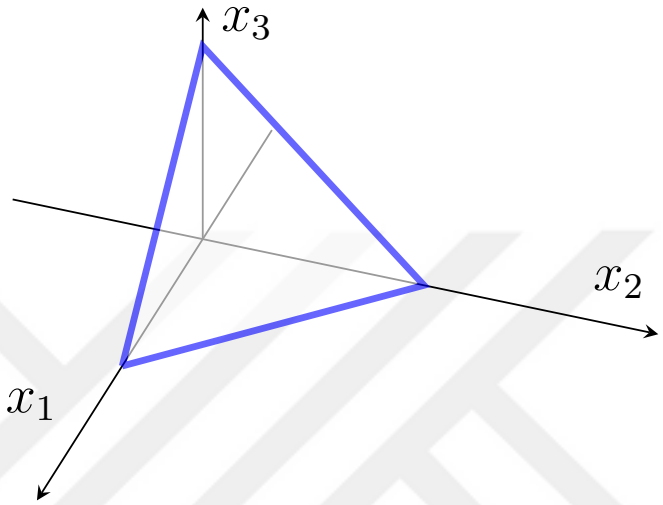

Figure 3.1: The 2-Sparse Probability Simplex on $\mathbb{R}^3$

By the nature of $\mathcal{P}$ and the unit simplex, an immediate observation follows.

Theorem 3.1. *Minimizing problem $\mathcal{P}$ is equivalent to solving problem*

$$\min_{S \subseteq I, |S|=k} \mathcal{P}(S),$$

where the subproblems $\mathcal{P}(S)$ are defined as

$$\begin{aligned} &\textit{minimize} && f(\mathbf{x}) \\ &\textit{subject to} && \mathbf{x} \in \Delta \\ &&& x_i = 0, \quad \forall i \in I \setminus S. \end{aligned} \tag{$\mathcal{P}(S)$}$$

and $\mathcal{P}^*(S)$ is the optimum value of a subproblem. That is,

$$\mathcal{P}^* = \min_{S \subseteq I, |S|=k} \mathcal{P}^*(S),$$

holds.

Proof. Let $\mathbf{x}^*$ be the optimum solution to $\mathcal{P}$. Then,

$$f(\mathbf{x}^*) \leq f(\mathbf{x}),$$

holds for every $\mathbf{x} \in \Delta_k$. In particular, for every set $S \subseteq I$ with $|S| = k$, we have that $S \subseteq \Delta_k$. Hence $f(\mathbf{x}^*) \leq f(\mathbf{x})$ holds for every $\mathbf{x} \in S$ and every $S \subseteq I$ with $|S| = k$. This gives $f(\mathbf{x}^*) = \mathcal{P}^* \leq \min\limits_{S \subseteq I, |S|=k} \mathcal{P}^*(S)$.

Conversely, assume that $S^* \in \arg\min\limits_{S \subseteq I, |S|=k} \mathcal{P}(S)$, where $\mathcal{P}(S)$ is defined as above and let $\mathbf{x}^* \in \arg\min \mathcal{P}$. Assume to the contrary that $\mathcal{P}^* = f(\mathbf{x}^*) < \min\limits_{S \subseteq I, |S|=k} \mathcal{P}^*(S)$. Since $\mathbf{x}$ is the optimum solution to $\mathcal{P}$, we must have $|I_1(\mathbf{x})| \leq k$. If $|I_1(\mathbf{x})| = k$, then by its definition we obtain $f(\mathbf{x}^*) = \mathcal{P}^*(I_1(\mathbf{x})) < \min\limits_{S \subseteq I, |S|=k}$, which is a contradiction since $I_1(\mathbf{x}) \subseteq I$ and $|I_1(\mathbf{x})| \leq k$. If $|I_1(\mathbf{x})| < k$, then consider any set with $I_1(\mathbf{x}) \subset T \subseteq I$ and $|T| = k$. Since $\delta^* \in \arg\min \mathcal{P}$, it follows that $\delta^* \in \arg\min \mathcal{P}(T)$. The same argument follows as in the equality case. Hence it must hold that $\mathcal{P}^* = f(\mathbf{x}^*) \geq \min\limits_{S \subseteq I, |S|=k} \mathcal{P}^*(S)$. Hence the equality

$$\mathcal{P}^* = \min_{S \subseteq I, |S|=k} \mathcal{P}^*(S),$$

holds. □

This reformulation implies that each subproblem solution for every k-sized subset of I is a candidate solution for optimality. Therefore, a natural condition for optimality follows from the subproblem characterization.

Definition 2. *Let $\mathbf{x} \in C$, and its support $I_1(\mathbf{x})$, $|I_1(\mathbf{x})| \leq k$. $\mathbf{x}$ is said to be* Basic Feasible *(BF) if it holds that*

$$\mathbf{x} \in \arg\min \mathcal{P}(S).$$

for every set $S \subseteq I$ such that $|S| = k$ and $I_1(\mathbf{x}) \subseteq S$.

In this definition, if $|I_1(\mathbf{x})| = k$, then the condition simplifies to

$$\mathbf{x} \in \arg\min \mathcal{P}(I_1(\mathbf{x})).$$

Checking the BF condition is an easy task for points with full support, as it only requires solving a convex subproblem. Checking the Basic Feasibility of a point with incomplete support might be more difficult if $|I_1(\mathbf{x})|$ is very small

compared to k, as this would require solving $\binom{k}{I_1(\mathbf{x})}$-many convex problems. Some well-known results about convex optimization are required to solve convex subproblems of the form S.

Fact 1 (Existence)**.** *A continuous function $f : C \longrightarrow \mathbb{R}$ on a compact set C attains its minimum.*

Fact 2 (Necessity)**.** *Consider minimizing a nonsmooth convex function f over C. Since f is convex, any local optimum is a global minimizer.*

Fact 3 (Sufficiency [60])**.** *Consider minimizing a real valued convex but nonsmooth function f over C. $\mathbf{x} \in C$ is a global optimum point of f if and only if there exists a $g \in \partial f(\mathbf{x})$ such that*

$$g^\intercal(\mathbf{y} - \mathbf{x}) \geq 0$$

for every $\mathbf{y} \in C$.

Together with Fact 2, these facts establish the necessary and sufficient conditions for optimality in the subproblem. The subproblem characterization gives a convenient way to define the BF condition, as one only needs to find a subgradient satisfying the sufficiency condition. Even though computing the subgradient is also an issue on its own, for the minimax case, subgradients are easy to identify. This fact is further exploited in the numerical experiments of Section 6.2.

Lemma 3.1. *Let $\mathbf{x} \in C$. $\mathbf{x}$ is* Basic Feasible *(BF) if and only if either*

- *It holds that $|I_1(\mathbf{x})| = k$ and there exists a subgradient $g \in \partial f^{I_1(\mathbf{x})}(\mathbf{x})$ such that*

  $$g^\intercal(\mathbf{y} - \mathbf{x}^{I_1(\mathbf{x})}) \geq 0$$

  for every $\mathbf{y} \in C^{I_1(\mathbf{x})}$.

- *It holds that $|I_1(\mathbf{x})| < k$ and there exists a subgradient $g \in \partial f^S(\mathbf{x})$ such that*

  $$g^\intercal(\mathbf{y} - \mathbf{x}^S) \geq 0$$

  for every $\mathbf{y} \in C^S$ and for every set $S \subseteq I$ such that $|S| = k$ and $I_1(\mathbf{x}) \subseteq S$.

Proof. Applying Facts 2 and 3 gives the desired result. □

It might not be practical to utilize the above conditions without exploiting the structure of f and C. The necessity and sufficiency are simplified significantly for special cases of C and f. To find optimality conditions specific to $\mathcal{P}$, consider the special case where C is chosen to be the n-dimensional unit simplex. The conditions for smooth optimization are given in [18], and the nonsmooth result should follow from there. However, the proof is included in this section for completeness.

Theorem 3.2 (Basic Feasible Characterization)**.** $\mathbf{x} \in \Delta$ *is a BF point with full support if and only if there exists a subgradient* $g \in \partial f(\mathbf{x})$ *and a* $\mu \in \mathbb{R}$ *such that*

$$g_i \begin{cases} = \mu & \mathbf{x}_i > 0 \\ \geq \mu & \mathbf{x}_i = 0. \end{cases}$$

$$\begin{aligned} g_i &= \mu \quad \textit{if} \quad \mathbf{x}_i > 0, \\ g_i &\geq \mu \quad \textit{if} \quad \mathbf{x}_i = 0. \end{aligned}$$

The incomplete support case follows similarly.

Proof. ($\implies$) Suppose $x \in \Delta$ satisfies the conditions in Lemma 3.1, with the corresponding subgradient g. There are two cases. If there are two indices i, j with $\mathbf{x}_i > 0$ and $\mathbf{x}_j > 0$, we can write that for a small δ, the point

$$\mathbf{x}^* = (\mathbf{x}_0, \mathbf{x}_1, \ldots \mathbf{x}_i + \delta, \ldots, \mathbf{x}_j - \delta, \ldots, x_n)$$

is trivially in the unit simplex. Therefore we can write the stationarity condition as

$$g^\intercal(\mathbf{x}^* - \mathbf{x}) \geq 0,$$

this gives

$$\delta g_i - \delta g_j \geq 0,$$

we get

$$g_i \geq g_j$$

changing δ to $-\delta$ yields the other side. Hence

$$g_i = g_j,$$

provided that both $\mathbf{x}_i > 0$ and $\mathbf{x}_j > 0$. Now it suffices to show that this holds when $\mathbf{x}_i > 0$ for some i and $\mathbf{x}_j = 0$ for $j \neq i$. For any such j now we can only write

$$\mathbf{x}^* = (\mathbf{x}_0, \mathbf{x}_1, \dots \mathbf{x}_i - \delta, \dots, \mathbf{x}_j + \delta, \dots, \mathbf{x}_n),$$

and this gives

$$-\delta g_i + \delta g_j \geq 0,$$

$$g_j \geq g_i.$$

Hence the subgradient elements corresponding to the support are all equal, while the others are greater than or equal to this number.

($\Longleftarrow$) Suppose now the claim holds. We can write

$$\begin{aligned} g^\intercal(\mathbf{x}_0 - \mathbf{x}) &= \sum_{i=1}^{n} g(x_0 - x)_i \\ &= \sum_{i=1}^{n} g_i(x_0)_i - \sum_{i=1}^{n} g_i x_i \\ &= \sum_{i=1}^{n} g_i(x_0)_i - \sum_{i:x_i>0} g_i x_i \\ &= \sum_{i=1}^{n} g_i(x_0)_i - \mu \\ &\geq \sum_{i=1}^{n} \mu(x_0)_i - \mu = 0, \end{aligned}$$

hence the condition in Lemma 3.1 is established. □

For the Minimax case there is a simpler characterization of subgradients.

Lemma 3.2 (max-Type Subgradients [59])**.** *Let $f : C \to \mathbb{R}$ be a pointwise maximum of convex functions $f_1, f_2, \dots, f_n$.*

$$f(x) = \max_{i \in \{1,\dots,n\}} f_i(x),$$

for every $x \in C$. Then its subdifferential at $x \in C$ is given by

$$\partial f(x) = \mathbf{Co}\left(\bigcup\{\nabla f_i(x) \mid f_i(x) = f(x)\}\right),$$

where $\mathbf{Co}(A)$ *denotes the convex hull of a set* A.

Therefore, given a point $\mathbf{x} \in \Delta$ one needs to solve the following linear system of equations with variables λ, μ to check this optimality condition for the subproblems of $\mathcal{P}$

$$\begin{aligned}
&\sum_{i \in D(\mathbf{x})} \lambda_i a_{ij} = \mu \qquad \forall j \in I_1(\mathbf{x}) \\
&\sum_{i \in D(\mathbf{x})} \lambda_i a_{ij} \geq \mu \qquad \forall j \in I_0(\mathbf{x}) \\
&\sum_{i \in D(\mathbf{x})} \lambda_i = 1 \\
&\lambda_i \geq 0,
\end{aligned}$$

where $D(\mathbf{x})$ is defined as follows:

$$D(\mathbf{x}) := \arg\max_{i \in I} a_i^\intercal \mathbf{x}.$$

BF points provide a sense of "local" optimality condition. Indeed, the interior of each feasible support is empty on $\mathbb{R}^n$, yet one can still talk about local optimality in the relative interior of each support. This notion is not well defined for points with incomplete support, but those can still be viewed as BF points of a problem with a stricter cardinality constraint.

Although useful, BF points are generally not globally optimal, and optimality is guaranteed only if one can extract the original optimum support and solve its corresponding subproblem. In fact, some BF points might be very bad candidates for the global optimum if they have incorrect support, especially in ill-posed examples. While convex subproblems can be solved efficiently, finding the correct support is generally computationally expensive. For problems having closed-form subproblem solutions, one can, in fact, find a better strategy to detect the optimum support. In problems without this property, such as $\mathcal{P}$, it is cumbersome to

compute the optimum support as one generally does not know the subproblem solutions in advance. To overcome this difficulty, searching for nearby supports instead of trying to detect the optimum support is a prominent attempt. Improving BF points with nearby supports is treated under the "Coordinatewise Optimum" points section.

3.3 Coordinatewise Optimality

For any point $\mathbf{x} \in \Delta_k$, by definition, at most k elements are allowed to be strictly positive. A convenient way of defining necessary optimality is that, one can compare $\mathbf{x}$ with vectors obtained by swapping its coordinates. The notion of comparison with coordinate swaps is provided as an extension of the definition of CW Optimality in [18].

Definition 3 (s-Neighbor)**.** *Given a point $\mathbf{x} \in C_k$ with support $I_1(\mathbf{x}) = k$, $\mathbf{y} \in C$ is said to be an s-Neighbor of $\mathbf{x}$ if there exists a permutation matrix P that differs from the identity matrix in at most s rows such that*

$$\mathbf{y} = P\mathbf{x}.$$

As an example, choose $n = 4, k = 3$ and $\mathbf{x} = (0.5, 0.3, 0.2, 0) \in \Delta_3^4$. The point $\mathbf{x}' = (0, 0.3, 0.2, 0.5)$ is a 2-neighbor of $\mathbf{x}$, because there exists

$$P = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

which differs from the identity matrix $\mathbf{I}$ in 2 rows and gives

$$P\mathbf{x} = \mathbf{x}'$$

A point that is better than all of its neighbors is said to be a *CW* point. The accuracy of this neighbor relationship is controlled through the permutations.

Definition 4 (s-CW Point)**.** *Given a point* $\mathbf{x} \in \Delta_k$ *with support* $I_1(\mathbf{x}) = k$*, we say that* $\mathbf{x}$ *is an s-CW point if*

$$\mathbf{x} \in \arg\min\{f(\mathbf{y}) \mid \mathbf{y} \in C, \mathbf{y} \text{ is an s-Neighbor of } \mathbf{x}\}.$$

The most natural special case of this definition is the 2-CW optimum points, which coincides with the Full-CW definition of [18]. For simplicity, 2-CW optimum points are called CW-points. Indeed, 2-CW points should be the computationally cheapest to check.

Starting from a Basic Feasible point, CW-optimality concept provides a way to search through neighboring supports. An algorithm to find CW points is not provided in this chapter, but one can search for CW points in a greedy-type technique. This qualification comes with an immediate drawback: Even if $\mathbf{x} \in \Delta_k$ is a point that is both BF and CW, the Basic Feasible points of a neighboring support might still be better than $\mathbf{x}$. This is still possible, as $\mathbf{x}$ can be BF and it only needs to be better than its swapped versions to be considered as a CW point. Unsurprisingly, combining BF and CW results and comparing $\mathbf{x}$ with the Basic Feasible points of neighboring supports directly yields a better optimality condition.

3.4 Neighbor Basic Feasibility

(2-)Coordinatewise Optimality can be a weak condition for nonsmooth problems. In this section, a stronger condition called *Neighbor Basic Feasibility* (NBF) is developed. Due to its construction, NBF is not significantly more expensive compared to finding CW points. Instead of searching over the permutations of vector $\mathbf{x}$, the idea is to simply focus on finding neighboring supports *better* than $I_1(\mathbf{x})$.

Definition 5 (Support Neighbor)**.** *For any point* $\mathbf{x} \in C$ *with* $|I_1(\mathbf{x})| = k$ *, a subset* $S \subseteq I$ *is a* support neighbor *of* $I_1(\mathbf{x})$ *if* $\exists i \in I_1(\mathbf{x}), j \in I_0(\mathbf{x})$ *such that*

$$S = (I_1(\mathbf{x}) \setminus \{i\}) \cup \{j\}.$$

The set of all possible support neighbors of $\mathbf{x}$ is denoted by $\mathcal{I}_{\mathbf{x}}$.

Definition 6. *A Basic Feasible point* $\mathbf{x}$ *with full support is called a Neighbor Basic Feasible Point (NBF) if*

$$f(\mathbf{x}) \leq \min_{S \in \mathcal{I}_{\mathbf{x}}} \mathcal{P}^*(S).$$

A BF point with incomplete support is declared to be NBF.

This is a stronger condition compared to the BF condition, and this condition is expected to be much more reliable than the BF and CW condition. Indeed, the following theorem shows that the set of NBF points is a subset of CW and BF points.

Theorem 3.3. *Let* $NBF_{\mathcal{P}}$*,* $BF_{\mathcal{P}}$ *and* $CW_{\mathcal{P}}$ *denote the set of NBF points, BF points and CW points of problem* $\mathcal{P}$*, respectively. Then it holds that*

$$NBF_{\mathcal{P}} \subseteq BF_{\mathcal{P}} \cap CW_{\mathcal{P}}$$

Proof. Let $\mathbf{x}$ be an NBF point to $\mathcal{P}$. Then by definition it is a BF point. One only needs to show that $\mathbf{x}$ is a CW point. For any S in the set $\mathcal{I}_{\mathbf{x}}$, we know that

$$f(\mathbf{x}) \leq \mathcal{P}^*(S)$$

by the definition of NBF points. Let P be any 2-permutation matrix as in the definition of CW points and let

$$\mathbf{y} = P\mathbf{x}.$$

Note that the support of $\mathbf{y}$ is determined by the permutation matrix P with at most two different rows from the identity matrix, and $\mathbf{y}$ is a 2-neighbor of $\mathbf{x}$. So it follows that $I_1(\mathbf{y}) \in \mathcal{I}_{\mathbf{x}}$. Hence

$$f(\mathbf{x}) \leq \mathcal{P}^*(I_1(\mathbf{y})),$$

and since $\mathbf{y}$ is a feasible point to the subproblem $P^(I_1(\mathbf{y}))$, we obtain

$$f(\mathbf{x}) \leq \mathcal{P}^*(I_1(\mathbf{y})) \leq f(\mathbf{y}).$$

Moreover, any 2-Neighbor of $\mathbf{x}$ is of this form and can be constructed with some permutation matrix P. Hence $\mathbf{x}$ is a CW point. □

Obivously, optimum solutions to $\mathcal{P}$ are NBF points. The conditions developed in this section, including the NBF, are computationally easy to check.

In summary, optimality conditions developed in this chapter form the backbone of the cardinality constrained minimax optimization problem $\mathcal{P}$. Even though none of these conditions provide a sufficient condition of optimality, they impose a sense of "local" optimality in an ill-posed domain. Consequently, algorithms and methods of subsequent chapters aim to find points satisfying these conditions.

3.5 Algorithms to Find Candidate Solutions

In this section, algorithms to find points satisfying BF, CW and NBF optimality conditions are presented. Basic Feasibility is the simplest form of the optimality conditions and can be solved through a convex (linear in this case) program. Given a feasible solution $\mathbf{x}$, an algorithm to find BF points is presented in Algorithm 1.

Algorithm 1 FindBF

1: Start with some $\mathbf{x} \in \Delta_k$ and compute $I_1(\mathbf{x})$.
2: Solve the convex program $\mathcal{P}(I_1(\mathbf{x}))$, obtain its minimizer $\mathbf{x}^*_{\mathbf{BF}}$.
3: Return $\mathbf{x}^*_{\mathbf{BF}}$

A useful strategy to approach $\mathcal{P}$ would be to find BF and CW points iteratively. To find BF-CW points, Algorithm 2 is presented. In this algorithm, FindCW($\mathbf{x}$) denotes a greedy CW finder algorithm. FindCW($\mathbf{x}$) can be a simple algorithm that checks 2-Neighbors of any feasible point arbitrarily and iteratively moves to better feasible solutions. Moreover, problem specific applications might be useful as one can improve FindCW($\mathbf{x}$) by choosing neighbors in a more efficient and reliable way. This algorithm is left as a design choice to the experimenter.

Unfortunately the accuracy of points determined by Algorithm 2 is not satisfactory even for relatively well-posed examples like $\mathcal{P}$. Due to Theorem 3.3, NBF

Algorithm 2 Greedy BF-CW Search

Start with some $\mathbf{x} \in C$ such that $I_1(\mathbf{x}) \leq k$.
Set BF =False and CW =False
while BF,CW = False,False **do**
  Call FindBF($\mathbf{x}$), an arbitrary subproblem solver.
  if $\mathbf{x} \in$ FindBF($\mathbf{x}$) **then**
    Set $BF = True$
  else
    $\mathbf{x} \longleftarrow$ FindBF($\mathbf{x}$)
  end if
  Call FindCW($\mathbf{x}$), an arbitrary CW-finder.
  if $\mathbf{x} \in$ FindCW($\mathbf{x}$) **then**
    Set $CW = True$
  else
    $\mathbf{x} \longleftarrow$ FindCW($\mathbf{x}$)
  end if
end while

point condition could be a better qualification technique. Starting from an initial BF point, Algorithm 3 searches for an NBF point in the neighboring supports in a greedy fashion.

Algorithm 3 NBF Point Search

Start with some BF point $\mathbf{x}$ with $|I_1(\mathbf{x})| = s$, Optimum = False.
while Optimum = False **do**
  Find the BF point on face $I_1(\mathbf{x})$: $x_{I_1(\mathbf{x})}$ and its objective value $z = f^{I_1(\mathbf{x})}(x_{I_1(\mathbf{x})})$.
  Compute $\mathcal{I}_{\mathbf{x}}$ through enumeration
  Check if there is an improving support $\mathcal{I}_{\mathbf{x}}$.
  if there is no improving S **then**
    Optimum = True.
    return $I_1(\mathbf{x})$, $x_{I_1(\mathbf{x})}$, z
  else
    Set $I_1(\mathbf{x}) \leftarrow S$.
  end if
end while

Algorithm 3 relies heavily on the selection of the initial support. When the initial point is chosen arbitrarily, the initial support could be distant from the optimum support, which results in either converging to a weak NBF point or converging very slowly. To overcome this, hard-thresholding the relaxation optimum solution is proposed in Algorithm 4, although no guarantee of performance is provided.

Algorithm 4 Initializer Selection

1: Solve $\mathcal{P}_{\mathbf{Rel}}$ and obtain $\mathbf{x}^*_{\mathbf{Rel}}$.
2: Choose the indices of the largest s elements of $x^*_{\mathbf{Rel}}$ by enumeration and form S.
3: Solve $\mathcal{P}^*(S) \rightarrow \mathbf{x}$
4: **return** $\mathbf{x}$

Algorithms 2 and 3 are based on greedy-type approaches. These algorithms require searching through neighbors of a point or a support. When the algorithms are limited to the simplest case of $2-$Neighbors, one only needs to check the BF condition for around $C(n, 2)$ times, which is computationally tractable. On the other hand, when n is very large (e.g., in the order of 10^5), the algorithms become significantly slower as the solution times of convex programs increase. The performances of Algorithm 2 and 3 are evaluated in Chapter 6 for various n, m and k values, and compared to the MILP upper bounds of the formulation in $\mathcal{P}_{\mathbf{MILP}}$.

Computing the optimum support in these algorithms is most of the time not possible. Based on the common techniques used in the literature, finding better supports to $\mathcal{P}$ can be attempted through regularization. The idea behind this formulation is that it puts a penalty to less sparse solutions, and searching for the correct support over a convex set might be more desirable algorithmically. In Chapter 4, the regularized problem and its relationship with $\mathcal{P}$ is analyzed.

Chapter 4

Finding Solutions Through Regularization

This chapter analyzes the sparsity regularized problem $\mathcal{P}_{\mathbf{Reg}}$ and show correspondences between the basic feasible solutions of $\mathcal{P}$ and the local minima of $\mathcal{P}_{\mathbf{Reg}}$. For any penalty parameter $\beta > 0$, the Cardinality Regularized Minimax Optimization Problem (CRMOP) is defined as follows.

$$\begin{array}{ll} \text{minimize} & \max\limits_{i \in M} a_i^{\intercal}\mathbf{x} + \beta\|\mathbf{x}\|_0 \\ \text{subject to} & \mathbf{x} \in \Delta. \end{array} \qquad (\mathcal{P}_{\mathbf{Reg}}(\beta))$$

An equivalent formulation to this problem is

$$\text{minimize} \quad \max_{i \in M} a_i^{\intercal}\mathbf{x} + \delta_{\Delta}(\mathbf{x}) + \beta\|\mathbf{x}\|_0, \qquad (P_{\mathrm{R}}(\beta))$$

where $\delta_C(\mathbf{x})$ is the indicator function on set $C \subseteq \mathbb{R}^n$,

$$\delta_C\mathbf{x}) = \begin{cases} 0 & \mathbf{x} \in C \\ +\infty & \mathbf{x} \in \mathbb{R}^n \setminus C. \end{cases} \qquad (4.1)$$

Since the domain C is a convex set, the function $f(\mathbf{x}) + \delta_C(\mathbf{x})$ is convex. However, both problems are nonconvex and discontinuous due to the cardinality constraint. Throughout this chapter, the following notation for the is used for the functions

$$f(x) = \max_{i \in M} a_i^T\mathbf{x},$$

$$f_\Delta(x) = \max_{i \in M} a_i^T \mathbf{x} + \delta_\Delta(\mathbf{x}),$$
$$F(x;\beta) = \max_{i \in M} a_i^T \mathbf{x} + \beta \|\mathbf{x}\|_0,$$
$$F_\Delta(x;\beta) = \max_{i \in M} a_i^T \mathbf{x} + \beta \|\mathbf{x}\|_0 + \delta_\Delta(\mathbf{x}),$$

where the indicator function is as in (4.1). A simple observation shows that problem $\mathcal{P}_{\mathbf{Reg}}(\beta)$ indeed has solutions. To see this, the following well-known fact is inherited.

Fact 4. *[61] A lower semicontinuous function $f(\mathbf{x})$ over a compact set C attains its minimum. That is, there exists a feasible solution $\mathbf{x}$ such that*

$$f(\mathbf{x}) = \min_{\mathbf{y} \in C} f(\mathbf{y})$$

Since Δ is compact, the lower semicontinuity of F establishes the existence of a global optimum.

Lemma 4.1. *For any $\mathbf{x}_0 \in \Delta$, there exists a radius $r_{\mathbf{x}_0}$ such that every $\mathbf{x} \in B_1(\mathbf{x}_0, r_{\mathbf{x}_0})$ has the property*

$$\|\mathbf{x}\|_0 \geq \|\mathbf{x}_0\|_0,$$

where $B_1(x, r)$ is the L_1 ball around point x with radius $r > 0$.

Proof. Choose $r_{\mathbf{x}_0} = \dfrac{\min\limits_{i \in I_1(\mathbf{x}_0)} \mathbf{x}_{0_i}}{2}$ and let $\mathbf{x} \in B(\mathbf{x}_0, r_{\mathbf{x}_0})$. Then, it holds that

$$\sum_{i=1}^{n} |\mathbf{x}_i - \mathbf{x}_{0_i}| \leq r \implies |\mathbf{x}_i - \mathbf{x}_{0_i}| \leq r, \quad \forall i$$

Then, this implies

$$\mathbf{x}_{0_i} - r \leq \mathbf{x}_i \leq r + \mathbf{x}_{0_i} \implies \mathbf{x}_i > 0, \forall i \in I_1(\mathbf{x}_0)$$

Hence $I_1(\mathbf{x}_0) \subseteq I_1(\mathbf{x})$ and the result follows. □

The result in Lemma 4.1 is quite similar to the result in [25], although the statement and the proof itself is slightly different in this version. Together with Lemma 4.1, now it is possible to show that F is lower semicontinuous.

Lemma 4.2. *The function $F(\cdot;\beta) : \Delta \to \mathbb{R}$ for any fixed β is lower semi-continuous.*

Proof. We need to show that

$$\liminf_{x \to x_0} F(x;\beta) \geq F(x_0;\beta)$$

for every $x_0 \in \Delta$. The statement is equivalent to showing

$$\sup_{\delta>0} \inf_{x \in B(x_0,\delta) \cap \Delta \setminus \{x_0\}} F(x;\beta) \geq F(x_0;\beta).$$

It can be shown that the infimum with respect to δ is a decreasing function. Moreover, one finds the supremum by decreasing the value of δ. Hence we have

$$\begin{aligned}
\sup_{\delta>0} \inf_{x \in B(x_0,\delta) \cap \Delta \setminus \{x_0\}} F(x;\beta) &= \sup_{\delta>0, \delta \leq r_{\mathbf{x}_0}} \inf_{x \in B(x_0,\delta) \cap \Delta \setminus \{x_0\}} F(x;\beta) \\
&= \sup_{\delta>0, \delta \leq r_{\mathbf{x}_0}} \inf_{x \in B(x_0,\delta) \cap \Delta \setminus \{x_0\}} \max_{i \in M} a_i^T x + \beta \|x\|_0 \\
(\text{Since } \delta \leq r_{\mathbf{x}_0}) &\geq \sup_{\delta>0, \delta \leq r_{\mathbf{x}_0}} \inf_{x \in B(x_0,\delta) \cap \Delta \setminus \{x_0\}} \max_{i \in M} a_i^T x + \beta \|x_0\|_0 \\
&= \beta \|x_0\|_0 + \liminf_{x \to x_0} f(x;\beta) \\
&= \beta \|x_0\|_0 + f(x_0;\beta) = F(x_0,\beta)
\end{aligned}$$

Hence the lower semi-continuity follows. □

Theorem 4.1. *The minimizer of $F(\mathbf{x};\beta)$ over Δ exists.*

Proof. By Lemma 4.2, $F(x;\beta)$ is lower semi-continuous on the probability simplex. Moreover, Δ is a compact set and thus, the minimizer of F over the simplex exists by Fact 4. □

Despite the fact that the global minimum exists, it is not possible to find the exact optimum solution in practice most of the time. Therefore, the aim of this chapter is to characterize the local optimum points of $\mathcal{P}_{\mathbf{Reg}}(\beta)$. The task of identifying the local minima requires various results and definitions, which will be provided in the next section.

4.1 Useful Definitions, Results and Theorems

The structural results related to the unit simplex and the minimax function are inherited to characterize the local minimums of $\mathcal{P}_{\mathbf{Reg}}(\beta)$. At any feasible point $\mathbf{x} \in \Delta$, the directional derivatives, and hence the local characteristics, are closely related to the maximizer indices of f. This concept needs to be formally defined before examining the local properties of directional derivatives. This is handled by defining the concepts of *degree* and *activity*.

Definition 7. *(Degree) For any point $\mathbf{x} \in \mathbb{R}_n$, we define its degree set by*

$$D(\mathbf{x}) = \{i \in M \mid a_i^T \mathbf{x} = f(\mathbf{x})\}.$$

One can immediately observe that for any $\mathbf{x} \in \mathbb{R}^n$, the degree set is nonempty.

Definition 8. *(Activity) For any subset of indices $S \subseteq M$, we define its activity by*

$$A(S) = \{\mathbf{x} \in \mathbb{R}^n \mid i \in D(\mathbf{x}), \forall i \in S\}.$$

Furthermore, if S is a single index, we shortly write $A(i)$ instead of $A(\{i\})$.

The concepts of *Activity* and *Degree* are closely related through the construction of $A(\cdot)$. These concepts are useful in the proofs, particularly in switching between feasible points and the subsets of the index set. The following lemma further strengthens this relationship.

Lemma 4.3. *For any $S \subseteq M$, $\mathbf{x} \in A(S)$ if and only if $S \subseteq D(\mathbf{x})$.*

Proof. Let $\mathbf{x} \in A(S)$ for some S. For any $k \in S$, we have that $k \in D(\mathbf{x})$ by the definition of $A(\cdot)$. Conversely suppose that $S \subseteq D(\mathbf{x})$. Then $i \in D(\mathbf{x})$ for every $i \in S$, so $\mathbf{x} \in A(S)$. □

Corollary 4.1. *If $D(\mathbf{x})$ is a singleton, then $\mathbf{x}$ is in the range of only one nonempty subset of M.*

Proof. Since $D(\mathbf{x})$ is a singleton, $\mathbf{x} \in A(S)$ and S must be either empty or equal to $D(\mathbf{x})$ by Lemma 4.3. By the definition of activity, if S is empty we have $A() =$ and $\mathbf{x} \notin A(S)$. Hence $S = D(\mathbf{x})$ and hence $\mathbf{x}$ is in the range of only one element. □

Lemma 4.4. *The activity sets are closed and convex sets.*

Proof. If $A(S) = \emptyset$ for $S \subseteq M$, the proof is trivial. Suppose $A(S) \neq \emptyset$.

(Convexity) Let $x, y \in A(S)$, $\lambda \in (0, 1)$. From 4.1, we have that $S \subseteq D(x)$ and $S \subseteq D(y)$. Consider the point $\lambda x + (1 - \lambda)y$ and let $k \in S$. By convexity of $f(\cdot)$,

$$\max_{i \in M} a_i^\intercal \left(\lambda x + (1 - \lambda)y\right) \leq \lambda \max_{i \in M} a_i^\intercal x + (1 - \lambda) \max_{i \in M} a_i^\intercal y.$$

Applying the definition of degree gives

$$\begin{aligned} f(\lambda x + (1 - \lambda)y) &= \max_{i \in M} a_i^\intercal \left(\lambda x + (1 - \lambda)y\right) \\ &\lambda \max_{i \in M} a_i^\intercal x + (1 - \lambda) \max_{i \in M} a_i^\intercal y \\ &= \lambda a_k^\intercal x + (1 - \lambda) a_k^\intercal y \\ &= a_k^\intercal \left(\lambda x + (1 - \lambda)y\right). \end{aligned}$$

We trivially have that

$$f(\lambda x + (1 - \lambda)y) \geq a_k^\intercal \left(\lambda x + (1 - \lambda)y\right),$$

so it follows that

$$f(\lambda x + (1 - \lambda)y) = a_k^\intercal \left(\lambda x + (1 - \lambda)y\right).$$

Hence $k \in D(\lambda x + (1 - \lambda)y)$ and $S \subset D(\lambda x + (1 - \lambda)y)$ is established. Again by 4.1, $\lambda x + (1 - \lambda)y \in A(S)$. Therefore $A(S)$ is convex.

(Closedness) Let $\{x_n\}_{n \in \mathbb{N}}$ be a sequence in $A(S)$ for some S converging to $x \in \mathbb{R}^n$. We have that $S \subseteq D(x_n)$, and we want to show that $S \subseteq D(x)$.

Let $k \in S$. This means that $k \in D(x_n)$ for each $n \in \mathbb{N}$. We get

$$a_k^\intercal x_n = \max_{i \in M} a_i^\intercal x_n.$$

By continuity of max function and the linear function, we get that

$$a_k^\intercal x - \max_{i \in M} a_i^\intercal x = \lim_{n \to \infty} \left(a_k^\intercal x_n - \max_{i \in M} a_i^\intercal x_n \right) = 0.$$

Hence

$$a_k^\intercal x = f(x),$$

which gives $k \in D(x)$. So $S \subseteq D(x)$ and $x \in A(S)$, which establishes closedness of $A(S)$.

□

The definitions of activity and degree allow one to investigate the local properties of $f(\mathbf{x})$ around any feasible point $\mathbf{x} \in \Delta$. In particular, there is a positive distance between $\mathbf{x}$ and the activity set of an index that is not contained in the degree of $\mathbf{x}$. Hence, dealing with local minima becomes significantly easier than considering the complete index set I around $\mathbf{x}$.

Lemma 4.5. *Let $\mathbf{x}_0 \in \mathbb{R}^n$. Then we have that*

$$d(\mathbf{x}_0, A(i)) = 0,$$

for every $i \in D(\mathbf{x}_0)$. Moreover,

$$d(\mathbf{x}_0, A(j)) > 0,$$

for every $j \in I \setminus D(\mathbf{x}_0)$, where $d(x, A)$ is the distance function between point x and set A, defined as

$$d(\mathbf{x}, A) := \inf_{\mathbf{y} \in A} d(\mathbf{x}, \mathbf{y})$$

and $d(\mathbf{x}, \mathbf{y})$ is any metric on the metric space.

Proof. Since $\mathbf{x}_0 \in A(i)$ for every $i \in D(\mathbf{x}_0)$, the first statement follows from the definition of the distance. Since $A(j)$ are closed sets by Remark 4.4, $j \notin D(\mathbf{x}_0)$, $\mathbf{x}_0 \notin A(j)$, and the set $\{\mathbf{x}_0\}$ is compact, it follows by Exercise 27.2 in [62] that

$$d(\mathbf{x}_0, A(j)) > 0$$

□

Using the above lemma, it is now possible to restrict attention to the degree set of a point while searching for local minima.

Theorem 4.2. *Let $\mathbf{x}_0 \in \mathbb{R}^n$. Then there exists $r > 0$ such that for every $\mathbf{x} \in B(\mathbf{x}_0, r)$, we have $D(\mathbf{x}) \subseteq D(\mathbf{x}_0)$.*

Proof. If $D(\mathbf{x}_0) = I$, there is nothing to prove as any radius r works. By Lemma 4.5, we know that the distance $d(\mathbf{x}_0, A(j))$ is strictly positive for any $j \notin D(\mathbf{x}_0)$. Hence we can choose

$$r := \min\{d(\mathbf{x}_0, A(j)) \mid j \notin D(\mathbf{x}_0)\}.$$

Since $D(\mathbf{x}_0) \neq I$, the set of distances is nonempty. Moreover, there can only be a finite number of positive distances. So, this minimum is well-defined. Note that for any $x \in B(\mathbf{x}_0, r)$, we have that $j \notin D(x)$ if $j \notin D(\mathbf{x}_0)$. Taking the contrapositive of this statement yields the desired result. □

The radius r is called the *support preserving radius* of $\mathbf{x}_0$, denoted as $r(\mathbf{x}_0)$. A useful corollary immediately follows, showing that we can always find a radius with the same support at differentiable points. This also implies that there is a ball around any differentiable point that contains differentiable points.

Corollary 4.2. *Assume $D(\mathbf{x}_0) = \{i\}$ for some $i \in M$. Then there exists $r > 0$ such that for every $\mathbf{x} \in B(\mathbf{x}_0, r)$, we have that $D(\mathbf{x}) = \{i\}$.*

It is now possible to establish the local optimality conditions of $\mathcal{P}_{\text{Reg}}(\beta)$.

4.2 Basic Feasibility Characterization of the Local Optima

Define the following two sets to use in the rest of this section.

$$\mathcal{S} := \{x \in \mathbb{R}^n \mid 1^\intercal x = 0\},$$

$$K(\omega) = \{x \in \mathbb{R}^n \mid x_i = 0, \forall i \in I \setminus \omega\},$$

where $\omega \subseteq I$.

Even though the function $f(\mathbf{x}) = \max_{i \in M} a_i^\intercal \mathbf{x}$ is not differentiable, its directional derivatives exist by convexity.

Lemma 4.6. *Consider the point $\mathbf{x}_0$. For any direction $d \in B(0, r(\mathbf{x}_0))$, we have that*

$$f'(\mathbf{x}_0; d) = \max_{i \in D(\mathbf{x}_0)} a_i^\intercal d.$$

Proof. For any d with $d \in B(0, r(\mathbf{x}_0))$, we know that $D(\mathbf{x}_0 + d) \subseteq D(\mathbf{x}_0)$. Therefore, there exists at least one index $j \in I$ such that $j \in D(\mathbf{x}_0 + d)$ and $j \in D(\mathbf{x}_0)$. Since $A(j)$ is convex, for any $\epsilon \in (0, 1)$, it holds that

$$(1 - \epsilon)\mathbf{x}_0 + \epsilon(\mathbf{x}_0 + d) = \mathbf{x}_0 + \epsilon d \in A(j).$$

Hence $j \in D(\mathbf{x}_0 + \epsilon d)$.

$$\frac{\max_{i \in M} a_i^\intercal (\mathbf{x}_0 + \epsilon d) - \max_{i \in M} a_i^\intercal \mathbf{x}_0}{\epsilon} = \frac{a_j^\intercal (\mathbf{x}_0 + \epsilon d) - a_j^\intercal \mathbf{x}_0}{\epsilon} = a_j^\intercal d.$$

This gives

$$f'(\mathbf{x}_0; d) = \lim_{\epsilon \searrow 0} \frac{\max_{i \in M} a_i^\intercal (\mathbf{x}_0 + \epsilon d) - \max_{i \in M} a_i^\intercal \mathbf{x}_0}{\epsilon} = a_j^\intercal d.$$

Now it is left to show that

$$j \in \arg\max_{i \in D(\mathbf{x}_0)} a_i^\intercal d.$$

Suppose to the contrary that there is some $k \in D(\mathbf{x}_0)$ and $k \neq j$ with

$$a_j^\intercal d < a_k^\intercal d.$$

Since $k \in D(\mathbf{x}_0)$ and $j \in D(\mathbf{x}_0)$, it holds that

$$a_k^\intercal \mathbf{x}_0 = a_j^\intercal \mathbf{x}_0,$$

inserting the strict inequality gives

$$a_j^\intercal (\mathbf{x}_0 + d) < a_k^\intercal (\mathbf{x}_0 + d),$$

which is a contradiction to the fact that $j \in D(\mathbf{x}_0 + d)$. So we get

$$f'(\mathbf{x}_0; d) = a_j^\intercal d = \max_{i \in D(\mathbf{x}_0)} a_i^\intercal d.$$

□

Now consider the parametric optimization problem defined as:

$$\begin{aligned} h(r) = \quad & \text{minimize} && \max_{i \in M} a_i^T \mathbf{x} \\ & \text{subject to} && 1^\intercal \mathbf{x} = 0, \\ & && \|\mathbf{x}\|_2 \leq r, \\ & && \mathbf{x} \in \mathbb{R}^n. \end{aligned}$$

Properties of this parametric optimization problem is crucial to show local optimality. For simplicity, define a minimizer of $h(r)$ as

$$\begin{aligned} d_r^* \in \quad & \arg\min && \max_{i \in M} a_i^T \mathbf{x} \\ & \text{subject to} && 1^\intercal \mathbf{x} = 0, \\ & && \|\mathbf{x}\|_2 \leq r, \\ & && \mathbf{x} \in \mathbb{R}^n. \end{aligned}$$

Obviously, this problem always has a solution as one can choose the feasible vector $\mathbf{0} \in \mathbb{R}^n$.

Remark 1. *For any $r > 0$, $h(r) \leq 0$.*

Proof. Notice that 0 is a feasible solution to $h(r)$ for any $r > 0$. Therefore the remark follows. □

Remark 2. *The function $h(r)$ is linear, that is, it holds that*

$$\lambda h(r) = h(\lambda r)$$

for any $\lambda > 0$ and

$$\lambda d_r^* = d_{\lambda r}^*.$$

Proof. We have

$$\begin{aligned} \lambda h(r) = \quad & \text{minimize} \quad && \lambda \max_{i \in M} a_i^T \mathbf{x} \\ & \text{subject to} \quad && 1^\intercal \mathbf{x} = 0, \\ & && \|\mathbf{x}\|_2 \leq r, \\ & && \mathbf{x} \in \mathbb{R}^n. \end{aligned}$$

Make a change of variables $\lambda \mathbf{x} = \mathbf{u}$ to get

$$\begin{aligned} \lambda h(r) = \quad & \text{minimize} \quad && \max_{i \in M} a_i^T \mathbf{u} \\ & \text{subject to} \quad && \tfrac{1^\intercal \mathbf{u}}{\lambda} = 0, \\ & && \|\mathbf{u}\|_2 \leq \lambda r, \\ & && \mathbf{u} \in \mathbb{R}^n, \end{aligned}$$

which gives

$$\begin{aligned} \lambda h(r) = \quad & \text{minimize} \quad && \max_{i \in M} a_i^T \mathbf{u} \\ & \text{subject to} \quad && 1^\intercal \mathbf{u} = 0, \\ & && \|\mathbf{u}\|_2 \leq \lambda r, \\ & && \mathbf{u} \in \mathbb{R}^n, \end{aligned}$$

which is equal to $h(\lambda r)$. The second equality also follows similarly. □

The following observation completes the construction by defining a radius embedding the local optimality. For any direction $d \in \mathcal{S}$, moving in the direction d preserves the existing support for sufficiently small d. In particular, if one chooses d such that $\|d\|_\infty < \min_{k \in \{1,\ldots,n\}} |\mathbf{x}_k|$, no element of $\mathbf{x} + d$ can be zero or negative. Therefore, one only has to worry about increasing support for local optimality. For any $\mathbf{x} \in \mathbb{R}^n$, define the following quantity as the radius.

$$\rho_{\mathbf{x}} = \min \left\{ \min_{k \in \{1,\ldots,n\}} |\mathbf{x}_k|, r(\mathbf{x}), \frac{\beta \left\| d^*_{r(\mathbf{x})} \right\|_2}{|h(r(\mathbf{x}))|}, d(\mathbf{x}, \Delta_n) \right\}, \tag{4.2}$$

provided that $h(r(\mathbf{x})) < 0$ and $d(\mathbf{x}, \Delta_n) > 0$. If one of them is equal to zero, one can immediately assign $+\infty$ to the corresponding minimization argument to define $\rho_{\mathbf{x}}$.

To prove local optimality, firstly, the attention is restricted to direction vectors from subspace $\mathcal{S}$, as they enforce $\mathbf{x} + d \notin \Delta$.

Lemma 4.7. *Let $\mathbf{x} \in \mathbb{R}^n$. For any vector $d \in B(0, \rho_{\mathbf{x}}) \cap \mathcal{S}^c$, it holds that*

$$F_\Delta(\mathbf{x} + d; \beta) \geq F_\Delta(\mathbf{x}; \beta).$$

Proof. If $\mathbf{x} \in \Delta_n$, then for any $d \in B(0, \rho_{\mathbf{x}}) \cap \mathcal{S}^c$ we get

$$1^\intercal(\mathbf{x} + d) = 1^\intercal \mathbf{x} + 1^\intercal d \neq 1.$$

This implies $\mathbf{x} + d \notin \Delta_n$, which makes $f(\mathbf{x} + d) = +\infty$ and inequality becomes trivial.

If $\mathbf{x} \notin \Delta_n$, then for any $d \in B(0, \rho_{\mathbf{x}}) \cap \mathcal{S}^c$ we have $\mathbf{x} + d \notin \Delta_n$ and the inequality is valid. □

Given two vectors $\mathbf{x}$ and d, the following inequality holds by the separation property of max function.

$$\max_{i \in M} a_i^\intercal(\mathbf{x} + d) \leq \max_{i \in M} a_i^\intercal \mathbf{x} + \max_{i \in M} a_i^\intercal d. \tag{4.3}$$

To see that this is true, for any finite index set I and $x_i, y_i \in \mathbb{R}$ for each $i \in I$, one has that

$$x_i + y_i \leq \max_i x_i + \max_i y_i,$$

holds for any i, and taking the maximum of the left hand-side over i gives the desired result. The other side of the inequality is not true in general. In fact, by choosing d accordingly, one can make the difference between the two sides substantial. On the other hand, it is possible to show that the equality holds locally if $\mathbf{x}$ and d are chosen appropriately. The following lemma establishes this local separation property and is crucial in proving local optimality.

Lemma 4.8. *Let $\mathbf{x} \in \mathbb{R}^n$. For any vector $d \in B(0, \rho_{\mathbf{x}})$, we have that*

$$\max_{i \in D(\mathbf{x})} a_i^\intercal(\mathbf{x} + d) = \max_{i \in D(\mathbf{x})} a_i^\intercal d + \max_{i \in D(\mathbf{x})} a_i^\intercal \mathbf{x}$$

Proof. By (4.3), the equivalence of separated terms holds. That is,

$$\max_{i \in D(\mathbf{x})} a_i^\intercal(\mathbf{x} + d) \leq \max_{i \in D(\mathbf{x})} a_i^\intercal \mathbf{x} + \max_{i \in D(\mathbf{x})} a_i^\intercal d.$$

Since $\rho_{\mathbf{x}} \leq r(\mathbf{x})$, we know that $D(\mathbf{x}+d) \subseteq D(\mathbf{x})$ holds by 4.2. Moreover,

$$\max_{i \in M} a_i^\intercal(\mathbf{x}+d) = \max_{i \in D(\mathbf{x})} a_i^\intercal(\mathbf{x}+d).$$

Then, by the directional derivative inequality we get

$$\max_{i \in D(\mathbf{x})} a_i^\intercal(\mathbf{x}+d) = \max_{i \in M} a_i^\intercal(\mathbf{x}+d) \geq \max_{i \in M} a_i^\intercal \mathbf{x} + \max_{i \in D(\mathbf{x})} a_i^\intercal d = \max_{i \in D(\mathbf{x})} a_i^\intercal \mathbf{x} + \max_{i \in D(\mathbf{x})} a_i^\intercal d.$$

This establishes that

$$\max_{i \in D(\mathbf{x})} a_i^\intercal(\mathbf{x}+d) = \max_{i \in D(\mathbf{x})} a_i^\intercal \mathbf{x} + \max_{i \in D(\mathbf{x})} a_i^\intercal d.$$

□

Now define the operator

$$\phi(x) := \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Lemma 4.9. *Let $\mathbf{x} \in \mathbb{R}^n \setminus \{0\}$. For any vector $d \in B(0, \rho_{\mathbf{x}})$, we have that*

$$\sum_{i \in I} \phi(x_i + d_i) = \sum_{i \in I_1(\mathbf{x})} \phi(x_i) + \sum_{i \in I_0(\mathbf{x})} \phi(d_i)$$

Proof. See the proof of Lemma 2.1 in [25]. □

Lemma 4.10. *Let $\mathbf{x} \in \mathbb{R}^n \setminus \{0\}$. For any vector $d \in B(0, \rho_{\mathbf{x}}) \cap K^c(I_1(\mathbf{x}))$, we have that*

$$F_\Delta(\mathbf{x}+d;\beta) = f_\Delta(\mathbf{x}+d) + \beta\|\mathbf{x}+d\|_0 \geq f_\Delta(\mathbf{x}) + \beta\|\mathbf{x}\|_0 = F_\Delta(\mathbf{x};\beta).$$

Proof. By Lemma 4.7 we may assume that $d \in \mathcal{S}$. If $\mathbf{x} \notin \Delta_n$, $\rho_{\mathbf{x}} \leq d(\mathbf{x}, \Delta_n)$ forces $\mathbf{x}+d \notin \Delta_n$, hence $+\infty \geq +\infty$ and the inequality is trivial. So, we may also assume $\mathbf{x} \in \Delta_n$ and $\mathbf{x}+d \in \Delta_n$. By 4.8, we have

$$\max_{i \in M} a_i^\intercal(\mathbf{x}+d) + \beta\|\mathbf{x}+d\|_0 = \max_{i \in D(\mathbf{x})} a_i^\intercal \mathbf{x} + \max_{i \in D(\mathbf{x})} a_i^\intercal d + \beta\|\mathbf{x}+d\|_0$$

by the previous lemma from [25] (and assuming that we change supports),

$$\underbrace{\max_{i\in M} a_i^\intercal(\mathbf{x}+d)+\delta_{\Delta_n}(\mathbf{x}+d)}_{f_\Delta(\mathbf{x}+d)}+\beta\|\mathbf{x}+d\|_0 = \max_{i\in D(\mathbf{x})} a_i^\intercal\mathbf{x}+\max_{i\in D(\mathbf{x})} a_i^\intercal d+\delta_{\Delta_n}(\mathbf{x})+\beta\|\mathbf{x}+d\|_0$$

$$= f_\Delta(\mathbf{x})+\beta\|\mathbf{x}\|_0+\max_{i\in D(\mathbf{x})} a_i^\intercal d+\beta\sum_{i\in I_0(\mathbf{x})}\phi(d_i).$$

Now, since $d\in K^c(I_1(\mathbf{x}))$, there is at least one index such that $i\in I_0(\mathbf{x})$ but $x_i>0$. Moreover since $d\in B(0,\rho_{\mathbf{x}})$, we have 4.9 and

$$\sum_{i\in I_0(\mathbf{x})}\phi(d_i)\geq 1$$

therefore we get

$$\underbrace{\max_{i\in M} a_i^\intercal(\mathbf{x}+d)+\delta_{\Delta_n}(\mathbf{x}+d)}_{f_\Delta(\mathbf{x}+d)}+\beta\|\mathbf{x}+d\|_0 = \max_{i\in D(\mathbf{x})} a_i^\intercal\mathbf{x}+\max_{i\in D(\mathbf{x})} a_i^\intercal d+\delta_{\Delta_n}(\mathbf{x})+\beta\|\mathbf{x}+d\|_0$$

$$= f_\Delta(\mathbf{x})+\beta\|\mathbf{x}\|_0+\max_{i\in D(\mathbf{x})} a_i^\intercal d+\beta\sum_{i\in I_0(\mathbf{x})}\phi(d_i)$$

$$\geq f_\Delta(\mathbf{x})+\beta\|\mathbf{x}\|_0+\max_{i\in D(\mathbf{x})} a_i^\intercal d+\beta$$

$$\geq f_\Delta(\mathbf{x})+\beta\|\mathbf{x}\|_0+\min_{d\in B(0,\rho_{\mathbf{x}})\cap\mathcal{S}}\max_{i\in D(\mathbf{x})} a_i^\intercal d+\beta$$

$$\geq f_\Delta(\mathbf{x})+\beta\|\mathbf{x}\|_0+h\left(\frac{\beta\left\|d^*_{r(\mathbf{x})}\right\|_2}{|h(r(\mathbf{x}))|}\right)+\beta$$

holds provided that $|h(r(\mathbf{x}))|<0$. Now by homogeneity

$$h\left(\frac{\beta\left\|d^*_{r(\mathbf{x})}\right\|_2}{|h(r(\mathbf{x}))|}\right)=\frac{\beta}{|h(r(\mathbf{x}))|}h\left(\left\|d^*_{r(\mathbf{x})}\right\|_2\right)=\frac{\beta}{|h(r(\mathbf{x}))|}h(r(\mathbf{x}))=-\beta.$$

Hence we get

$$f_\Delta(\mathbf{x}+d)+\beta\|\mathbf{x}+d\|_0\geq f_\Delta(\mathbf{x})+\beta\|\mathbf{x}\|_0.$$

For the case with $h(r(\mathbf{x}))=0$, it is guaranteed that

$$\max_{i\in D(\mathbf{x})} a_i^\intercal d+\beta\geq 0,$$

for any d within given radius. So the result immediately follows. □

Using the above lemmas, the local optimality characterization is proven in cases.

Theorem 4.3 (Local Optimality)**.** *Let* $\mathbf{x} \in \Delta$*. For any vector* $d \in B(0, \rho)$*,* $\mathbf{x}$ *is a local optimum if and only if it is a solution to* $\mathcal{P}(I_1(\mathbf{x}))$*, which is equivalent to the Basic Feasibility condition.*

Proof. Assume that $\mathbf{x}$ is a BF point. We have shown that for any $\mathbf{x} \in \Delta$:

- Lemma 4.7: For any vector $d \in B(0, \rho) \cap \mathcal{S}^c$, it holds that

$$f(x + d) + \beta\|x + d\|_0 \geq f(x) + \beta\|x\|_0.$$

- Lemma 4.10: For any vector $d \in B(0, \rho) \cap K^c(I_1(x)) \cap \mathcal{S}$, we have that

$$f(x + d) + \beta\|x + d\|_0 \geq f(x) + \beta\|x\|_0.$$

So it suffices to show that we have the same statement hold for any $d \in B(0, \rho) \cap K(I_1(x)) \cap \mathcal{S}$. Note that since $d \in K(I_1(x))$ and $\|x\| \leq \min_{k \in \{1,\ldots,n\}} |\mathbf{x}_k|$ we get

$$\|x + d\|_0 = \|x\|_0,$$

Hence it follows that

$$I_1(\mathbf{x} + d) = I_1(\mathbf{x}).$$

In this case, if $\mathbf{x}+d \notin \Delta$, then the inequality is trivial since $+\infty \geq 0$. If $\mathbf{x}+d \in \Delta$, then one has

$$f(x + d) \geq f(x),$$

by the basic feasibility of $\mathbf{x}$. Therefore, we must have

$$f(x + d) + \beta\|x + d\|_0 \geq f(x) + \beta\|x\|_0,$$

for every $\mathbf{x} \in B(0, \rho_{\mathbf{x}})$.

Conversely, assume that $\mathbf{x}$ is a local minimum of $F_\Delta(\mathbf{x}; \beta)$ around the ball $B(0, \rho_{\mathbf{x}})$. Assume to the contrary that $\mathbf{x}$ is not a BF point and the optimum

solution of subproblem $\mathcal{P}(I_1(\mathbf{x}))$ is point $\mathbf{y}$. Then, choose $d_n = \dfrac{\mathbf{y}-\mathbf{x}}{n}$. Note that we must have

$$I_1(\mathbf{x}) = I_1(\mathbf{y})$$

and hence

$$I_1(\mathbf{x}) = I_1(\mathbf{y}) \supseteq I_1\left(\mathbf{x} + \frac{\mathbf{y}-\mathbf{x}}{n}\right).$$

One can choose n large so that $d_n \in B(0, \rho_{\mathbf{x}})$. Since $\mathbf{y}$ is the BF point and

$$f(\mathbf{y}) < f(\mathbf{x}),$$

by convexity of Δ, $\mathbf{x} + \frac{\mathbf{y}-\mathbf{x}}{n} \in \Delta$, and moreover by convexity of f,

$$f(\mathbf{y}) \leq f\left(\mathbf{x} + \frac{\mathbf{y}-\mathbf{x}}{n}\right) < f(\mathbf{x}),$$

which gives

$$F_\Delta\left(\mathbf{x} + \frac{\mathbf{y}-\mathbf{x}}{n}\right) < F_\Delta\left(\mathbf{x}\right),$$

which is a contradiction to the local optimality of $\mathbf{x}$. Hence $\mathbf{x}$ must be a BF point. □

By Theorem 4.3, it is now possible to see the parallels between $\mathcal{P}$ and $\mathcal{P}_{\mathbf{Reg}}$ clearly. Every BF point of $\mathcal{P}$ corresponds to a local optimum of $\mathcal{P}_{\mathbf{Reg}}$. Hence, one can solve the regularized problem for a good local optimum point to obtain a BF point of better quality. The advantage of $\mathcal{P}_{\mathbf{Reg}}$ is that it has a convex domain, meaning that it is possible to use well-studied methods such as DC programming, as well as methods utilizing projections to the convex domain.

While this problem has advantages both in theory and practice, the use of the regularized problem comes with a significant drawback, as its objective function is now discontinuous and non-convex. In 5, we introduce a novel surrogate to handle discontinuity of $\|\mathbf{x}\|_0$, and we propose a DC-based algorithm to find supports to $\mathcal{P}$.

Chapter 5

Finding Local Optima of the Regularized Problem

The connection between the regularized problem and the constrained problem is established in Chapter 4. Extracting the optimum solution of $\mathcal{P}$ from $\mathcal{P}_{\mathbf{Reg}}(\beta)$ requires two important steps: One first needs to determine the correct parameter β, and solve $\mathcal{P}_{\mathbf{Reg}}(\beta)$ efficiently. In practice, correct β values can be found by trial and error, provided a fast algorithm exists to solve the regularized problem. Unfortunately, no methods have been developed to solve $\mathcal{P}_{\mathbf{Reg}}(\beta)$-like problems exactly, and most of the methods in the literature rely on concave approximations of the cardinality penalty. In other words, instead of solving $\mathcal{P}_{\mathbf{Reg}}(\beta)$ directly, a practical approach is to solve an approximation of it, such as the problem

$$\begin{array}{ll} \text{minimize} & \max\limits_{i\in M} a_i^{\intercal}\mathbf{x} + \beta g(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in \Delta, \end{array} \qquad (\mathcal{P}_g(\beta))$$

where $g(\mathbf{x}) : \Delta^n \to \mathbb{R}$ is a concave and differentiable (usually, $g \in C^2$) penalty over the unit simplex. It is natural to assume that function g satisfies some properties, such as concavity and $g(\mathbf{0}) = 0$, and it promotes sparsity when solved. Concavity is mostly desired to replicate the function $\|\mathbf{x}\|_0$, although choosing g as a concave function results in a nonconvex program.

5.1 Difference of Convex Approach

Solving the approximation $\mathcal{P}_g(\beta)$ is, in general, significantly easier than $\mathcal{P}_{\mathbf{Reg}}(\beta)$ because the former is continuous and, most importantly, differentiable, whereas the latter is discontinuous. Among the methods that solve problems similar to $\mathcal{P}_g(\beta)$, perhaps one of the simplest approaches is the *Difference of Convex Algorithm* (DCA). For any problem of the form

$$\min_{\mathbf{x}\in C} f(\mathbf{x}) - h(\mathbf{x}), \tag{5.1}$$

where $f, h : C \to \mathbb{R}$ convex and differentiable functions on convex set C. The DCA in this thesis is used as per the definition in [63]. At each iteration k, the DCA solves the convex problem

$$\mathbf{x}_{k+1} \in \arg\min_{\mathbf{x}\in C} f(\mathbf{x}) - [h(\mathbf{x}_k) + \nabla h(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k)], \tag{5.2}$$

until convergence, typically $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| < \varepsilon$ for some threshold $\varepsilon > 0$. DCA is straightforward to implement and is specifically designed for problems of the form 5.1. Problem $\mathcal{P}_g(\beta)$ naturally has this property and is amenable to DCA, as $\beta > 0$ and the penalty is usually concave; hence, $-g$ is convex. The complete algorithm is given in Algorithm 5 for the case $\mathcal{P}_g(\beta)$.

Algorithm 5 Iterative Difference of Convex Algorithm

1: **Initialize** $\mathbf{x}_0 \in \Delta$, $\beta \in \mathbb{R}$, and tolerance ε
2: **while** $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| \geq \varepsilon$ **do**
3: Solve the following problem:

$$\mathbf{x}_{k+1} \in \arg\min_{\mathbf{x}\in\Delta} \max_i \mathbf{a}_i^\top \mathbf{x} + \beta g(\mathbf{x}_k) + \beta \nabla g(\mathbf{x}_k)^\top (\mathbf{x} - \mathbf{x}_k)$$

4: **end while**
5: **Output:** $\mathbf{x}_{k+1}$

In Sparse Optimization, the issue with Algorithm 5 is that it might not yield a desired sparsity level if the penalty β is not chosen appropriately. On the other hand, the algorithm solves minimax optimization problems at each iteration.

Hence, it is not very expensive time-wise. To recover k-sparse solutions, a useful strategy would be to increase β if the solution is *dense* and to decrease β if the solution of the DC algorithm is very sparse. In our experiments, a fixed sparsity level k might not be achieved exactly, no matter how carefully the penalty parameter β is chosen. To overcome this issue, Algorithm 6 is proposed instead, guaranteeing a k-sparse support. Even though this algorithm always outputs full support, one still requires function g to be sparsity-promoting. Thus, the success of Algorithm 6 relies heavily on selecting a correct concave regularizer function g.

Algorithm 6 k-Sparse Difference of Convex Algorithm

Initialize $\mathbf{x}_0 \in \Delta$, $\beta \in \mathbb{R}$, sparsity level $k < n$, and tolerance ε
while $\|\mathbf{x}_{l+1} - \mathbf{x}_l\| \geq \varepsilon$ **and** $\|\mathbf{x}_l\|_0 > k$ **do**
    Solve the following problem:

$$\mathbf{x}_{l+1} \in \arg\min_{\mathbf{x}\in\Delta} \max_i \mathbf{a}_i^\top \mathbf{x} + \beta g(\mathbf{x}_l) + \beta \nabla g(\mathbf{x}_l)^\top (\mathbf{x} - \mathbf{x}_l)$$

end while
Let $\mathbf{x}_l^*$ and $\mathbf{x}_{l+1}^*$ be the outputs of this loop.
Obtain $S \subseteq I$ be the indices of the k largest entries of $\mathbf{x}_l^*$. Solve $\mathcal{P}(S)$ and obtain $\mathbf{x}_S^*$.
if $k > |I_1(\mathbf{x}_{l+1})|$ **then**
    Obtain $T \subseteq I$ with the following procedure: Find $k - |I_1(\mathbf{x}_{l+1})|$-many largest elements of $\mathbf{x}_l^*$ that are not in $I_1(\mathbf{x})$. Let I' be this set. Then set $T := I_1(\mathbf{x}) \cup I'$. Solve $\mathcal{P}(T)$ and obtain $\mathbf{x}_T^*$
else
    Set $f(\mathbf{x}_T^*) = +\infty$
end if
Output $\mathbf{x} \in \arg\min\{f(\mathbf{x}_S^*), f(\mathbf{x}_T^*)\}$

In practice, finding a more accurate β is still an important task, and the parameter values closer to the ones that induce exact k sparsity most of the time yield better results than others. An interesting observation for this special case

is that at any iteration l, one can rearrange the terms in the inner optimization problem as follows

$$\mathbf{x}_{l+1} \in \arg\min_{\mathbf{x}\in\Delta} \max_i (\mathbf{a}_i + \beta\nabla g(\mathbf{x}_l))^\top \mathbf{x} + \beta g(\mathbf{x}_l) - \beta\nabla g(\mathbf{x}_l)^\top \mathbf{x}_l.$$

Since the $\mathbf{x}_l$ terms are constant, the optimization at each iteration reduces to an altered version of the original Minimax problem. Based on the gradient information of the penalty g at any point $\mathbf{x}$, the game matrix $\mathbf{A}$ columns are altered altogether. For instance, if at one index $j \in \{1, 2, \ldots, n\}$ the partial derivative of g at that index is large, then the column is penalized in the next iteration. On the other hand, if the partial derivative of g in the direction of j is small, then the variable of index j is prioritized. Based on this observation, the Concave Power Function Penalty (CPFP) is proposed.

5.2 Concave Power Function Penalty and Its Properties

The Concave Power Function Penalty (CPFP) is defined as follows:

$$g(\mathbf{x}; \alpha, \gamma) = \sum_{i=1}^{n} g_i(\mathbf{x}) = \sum_{i=1}^{n} x_i^\alpha (1 - x_i)^\gamma, \tag{5.3}$$

for some $\alpha \in (0, 1)$ and $\gamma \in (0, 1)$.

Lemma 5.1. *The CPFP is concave.*

Proof. For a fixed i, the second derivative of each term in the summation is

$$\alpha x_i^{\alpha-1}(1 - x_i)^\gamma - \gamma x_i^\alpha (1 - x_i)^{\gamma-1},$$

and the second derivatives are

$$\alpha(\alpha - 1)\mathbf{x}_i^{\alpha-2}(1 - \mathbf{x}_i)^\gamma - 2\alpha\gamma\,\mathbf{x}_i^{\alpha-1}(1 - \mathbf{x}_i)^{\gamma-1} + \gamma(\gamma - 1)\mathbf{x}_i^\alpha(1 - \mathbf{x}_i)^{\gamma-2}.$$

Since $\alpha, \gamma > 0$ and $1 - \alpha, 1 - \gamma < 0$, all terms are negative, hence by the second order characterization of concavity (conveity) [64], each term in CPFP is concave.

Since CPFP is a sum of concave functions, it is a concave function over the unit simplex [64]. □

The partial derivatives of $g(\mathbf{x}; \alpha, \gamma)$ are of the form

$$\frac{\partial g(\mathbf{x})}{\partial \mathbf{x}_j} = \alpha \mathbf{x}_j^{\alpha-1}(1-\mathbf{x}_j)^{\gamma} - \gamma \mathbf{x}_j^{\alpha}(1-\mathbf{x}_j)^{\gamma-1}. \tag{5.4}$$

This expression is well defined only if $\mathbf{x}_i \in (0, 1)$. For completeness, one can assign a very large number and a very small number to the partial derivative at 0 and 1, respectively. One can see that the penalty (5.3) promotes sparsity at each iteration of Algorithm 6 because it penalizes the values that are close to zero by increasing its column value at game matrix $\mathbf{A}$ significantly, at the same time, it promotes values that are closer to 1, by decreasing their column value in game matrix $\mathbf{A}$. Figure 5.1 illustrates the partial derivatives.

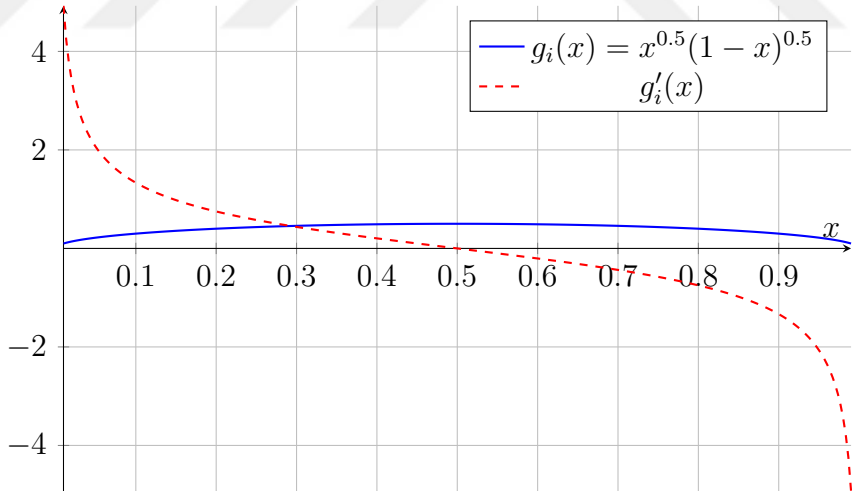

Figure 5.1: Plot of $g_i(x) = x^{0.5}(1-x)^{0.5}$ and its derivative $g_i'(x)$.

Note that the penalty $\beta \in \mathbb{R}_{++}$ effectively determines how sharp the penalties/rewards are assigned at each iteration. In particular, a small β value would correspond to a more *dense* solution in Algorithm 5 because the function is rather flat for most values of x_i. Moreover, increasing β should make the solution more sparse, as the penalty/reward is increased even for values of $\mathbf{x}$ far away from the boundaries of Δ. (See Figure 5.2)

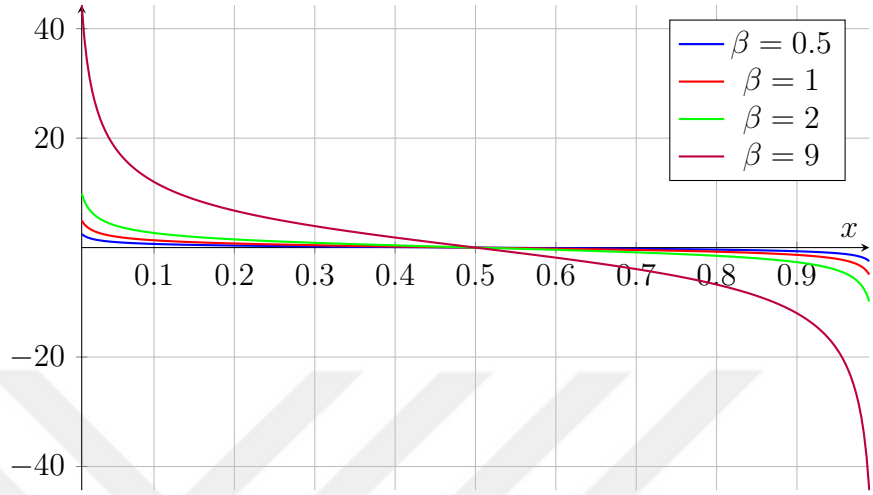

Figure 5.2: Plot of $\beta g_i'(x)$ for different values of β,$\alpha = \gamma = 0.5$.

Based on the column-altering interpretation of the DCA, penalty function 5.3 satisfies all required properties. At the same time, this penalty promotes elements that have a value greater than 0.5 even more, resulting in sparser solutions.

5.3 Parameter Selection

Function parameters in (5.3) can be changed based on the dynamics of the game matrix $\mathbf{A}$. For example, it is natural to expect that when n and k become very large, the elements of the feasible solutions could become very small. In this case, this might affect the way Algorithm 6 terminates shortly after several iterations due to the zeros being demoted and positive values being promoted.

If the algorithm terminates with the condition $\|\mathbf{x}_l\|_0 \leq k$, one might consider decreasing β values to achieve a better solution and increase β otherwise. A procedure to determine better β values is given in Algorithm 7.

Algorithm 7 β Determination Procedure

Initialize $\beta_{min} = 0$, $\beta_{max} = M$ for some large M, sparsity level $k < n$, tolerance $\tau \in \mathbb{N}$ and iteration limit $N \in \mathbb{N}$
while $|\|\mathbf{x}\|_0 - k| \geq \tau$ **and** $l < N$ **do**
  Solve Algorithm 6 with $\beta_l = \frac{\beta_{min}+\beta_{max}}{2}$ and obtain best solution $\mathbf{x}^*$ and $\|\mathbf{x}^*\|_0$.
  if $\|\mathbf{x}^*\|_0 < k$ **then**
    Set $\beta_{max} \leftarrow \beta_l$
  else if $\|\mathbf{x}^*\|_0 > k$ **then**
    Set $\beta_{min} \leftarrow \beta_l$
  else
    Break
  end if
  Set $l \leftarrow l + 1$.
end while
Output β_l

Selecting values for α and γ determines the maximum point of the function (5.3). The simplest case is, perhaps, choosing $\alpha = \gamma = 0.5$.

Lemma 5.2. *The maximizer of penalty in* (5.3) *with* $\alpha = \gamma = 0.5$ *is the point*

$$\mathbf{x}^* = \left(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}\right). \tag{5.5}$$

Proof. For any point with $\mathbf{x}_i = 1$, the objective value is trivially 0. On the other hand, by Theorem 3.2, optimization over the unit simplex requires all positive values to have the same gradient value. Obviously, point (5.5) satisfies this condition. Hence it is a local optimum and by concavity of g, $\mathbf{x}^*$ is the maximizer over the unit simplex. □

Changing the maximum point of (5.3) can be useful if there is some prior knowledge of the optimum solution is present; this way, the parameters could be adjusted so that the algorithm terminates in a shorter amount of time. In the

extreme cases, when $\gamma \to 0$, CPFP gives

$$\sum_{i=1}^{n} x_i^{\alpha}, \tag{5.6}$$

and the penalty becomes very similar to the well known L_p norm penalty with $p \in (0, 1)$. Besides, if $\alpha, \gamma \to 1$, 5.3 gives

$$\sum_{i=1}^{n} x_i - x_i^2 = \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} x_i^2 = 1 - \|\mathbf{x}\|_2^2, \tag{5.7}$$

which is analogous to the negative L_2 norm, provided that $\mathbf{x} \in \Delta$. Therefore, this family of functions carries the properties of sparsity-promoting penalties common in the literature.

To assess the sensitivity of the parameters in the DCA, experiments were run with $m = n = 100, 150, 200, 1000$ and $m = 150, n = 200$ with $\beta = 10, 1, 1, 10, 10$, respectively, and sparsity values $k = 30$. The values for β and k are chosen arbitrarily; for the experiments with $m = n = 100, 150, 200$ and $m = 150, n = 200$, 12 equally separated values from the interval $(0, 1)$ were selected for α and γ, hence in total 144 values were compared for each experiment. For $m = n = 1000$, 13 values from the same interval were chosen, a total of 169 values for α, γ pairs. The case with $n = m = 1000$, $k = 30$, $\beta = 10$ can be found in Figures 5.3 and 5.4. The rest of the experiments for other m, n values can be found in Appendix A. Table 5.1 shows the best parameter selections for the DCA in terms of the objective values among all experiments. As an example, for the instance with a 1000×1000 matrix, the best parameters were seen to be approximately $(\alpha, \gamma) = (0.64, 0.36)$.

The analysis shows that choosing $\alpha \approx 0.6$ or $\alpha \approx 0.7$ most often gives the best results. On the other hand, the solutions can be improved via altering γ in some examples. This essentially means that by altering γ, one can obtain better solutions than the regular L_p norm with $p \in (0, 1)$. Furthermore, most of the time, solutions are better than the negative L_2 norm penalty.

For any selection of α, γ pairs, the structure of (5.3) makes partial derivatives at 0 very large. From a practical standpoint, this observation is useful because

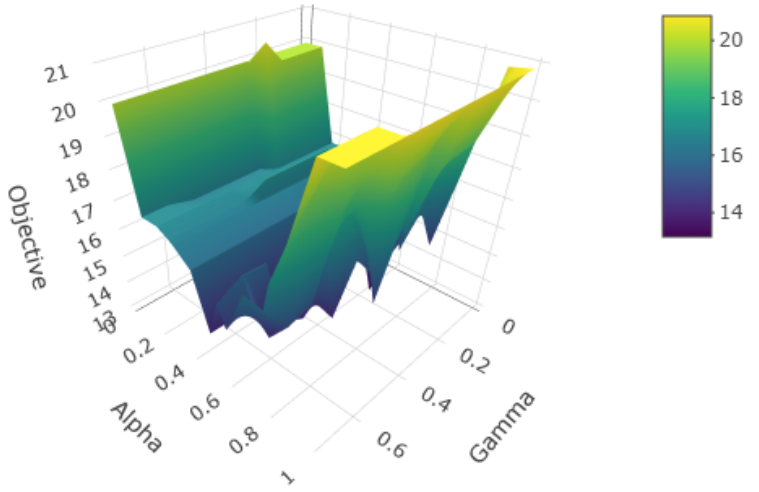

Figure 5.3: Analysis of CPFP α and γ Parameters, 1000×1000 Sample

once a variable is set to 0 in an iteration, it is penalized severely and should not be selected in the following iteration. This way, the support at each iteration should decrease, and it is possible to extract the same results by solving smaller subproblems. As a result, the DCA approach in Algorithm 6 can be modified for faster convergence.

5.4 Modified DC Approach with CPFP

When applying Algorithm 6, consider switching ∇g with $\mathcal{G}$. This function essentially mimics the original penalty 5.3, but is only different in the extreme values. This way, the gradient is computable for values at 0 and 1.

$$\mathcal{G}_j(\mathbf{x};\alpha,\gamma) := \begin{cases} -\min\limits_k a_{kj} + \max\limits_{(i,k)} a_{ik} + \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}_j}\big|_{x_j=10^{-n}} & \text{if } x_j \leq 10^{-n} \\ +\max\limits_k a_{kj} - \min\limits_{(i,j)} a_{ij} + \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}_j}\big|_{x_j=1-10^{-n}} & \text{if } x_j \geq 1-10^{-n} \\ \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}_j} & x_j \in (10^{-n}, 1-10^{-n}). \end{cases} \tag{5.8}$$

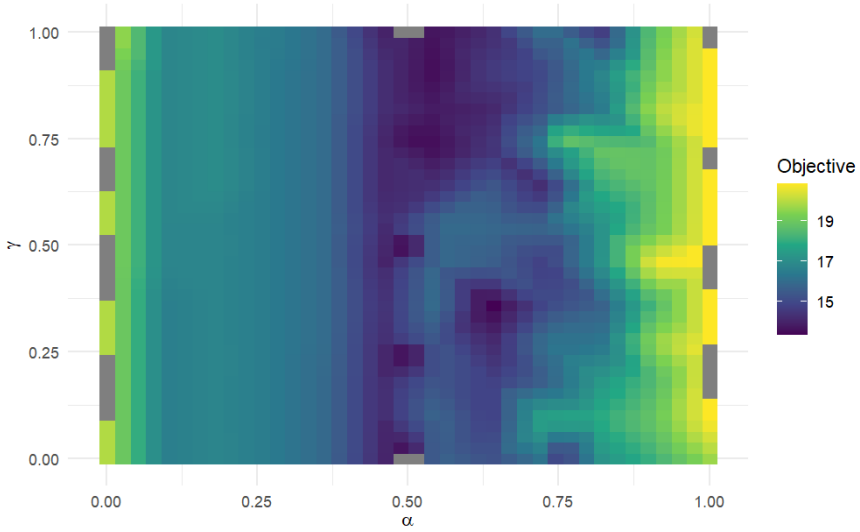

Figure 5.4: Surface Contours: CPFP α and γ Parameters, 1000×1000 Sample

This guarantees the algorithm to terminate at any solution involving any corner entry. Replacing the partial derivatives with (5.8) comes with several other benefits. In fact, one can simplify Algorithm 6 significantly using the properties of $\mathcal{G}$. This simplification is a special result of the original problem $\mathcal{P}$.

Lemma 5.3. *Let A be the game matrix of $\mathcal{P}$ and $\mathbf{x}^*$ be the optimum solution of this problem. For a column $j \in \{1, \ldots, n\}$, if there exists a $t \neq j$ such that*

$$\min_{i \in M} a_{ij} > \max_{u \in M} a_{ut}, \tag{5.9}$$

then $\mathbf{x}_j^* = 0$.

Proof. Assume that 5.9 holds and j, t are the columns with this property. Assume that $\mathbf{x}_j^* = \epsilon$ for some $\varepsilon > 0$, and $f(\mathbf{x}^*)$ is the optimum solution. We have that

$$f(\mathbf{x}^*) = a_{ij}\varepsilon + a_{it}\mathbf{x}_t^* + \sum_{k \neq j,t} a_{ik}\mathbf{x}_k^*,$$

holds for every $i \in D(\mathbf{x})$ by the definition of degree set. Notice that due to the property in 5.9, we must have

$$a_{ij} > a_{it},$$

for every $i \in M$. So we get that

$$a_{ij}\varepsilon + a_{it}\mathbf{x}_t^* > a_{it}(\mathbf{x}_t^* + \varepsilon),$$

m	n	Best α	Best γ
1000	1000	0.63636	0.36364
100	100	0.63636	0.00001
		0.63636	0.09092
		0.63636	0.18182
		0.63636	0.27273
		0.63636	0.36364
		0.63636	0.45455
150	200	0.72727	0.90908
150	150	0.63636	0.00001
		0.63636	0.09092
		0.63636	0.18182
		0.63636	0.27273
		0.63636	0.36364
		0.63636	0.45455
		0.63636	0.54545
		0.63636	0.63636
		0.63636	0.72727
		0.63636	0.81818
		0.63636	0.90908
		0.63636	0.99999
200	200	0.72727	0.36364

Table 5.1: Best (α, γ) Pairs in Experiments

for every $i \in M$ and t. Therefore, switching ε from column j to t reduces the inner product for every row $i \in M$, including but not up to the rows in $D(\mathbf{x})$. Therefore, the new vector has an optimum solution less than $f(\mathbf{x}^*)$, which is a contradiction to the optimality of $\mathbf{x}^*$. So $\mathbf{x}^*_j = 0$ must hold. □

Theorem 5.1. *Let l be any iteration of the 6 and $\mathbf{x}^l_j$ is the last solution obtained at l. Moreover, assume that the gradient is replaced with $\mathcal{G}$, where $\mathcal{G}$ denote the vector of partial derivatives defined as 5.8. If $x^l_j \leq 10^{-n}$, then it holds that $\mathbf{x}^{l+1}_j = 0$. In particular, if $\mathbf{x}^l_j = 0$, then $\mathbf{x}^{l+1}_j = 0$.*

Proof. It suffices to show that the new matrix at iteration $l + 1$ satisfies the property in 5.9. Let $\mathbf{x}^l_j < 10^{-n}$. An important observation here is that, there must be an index k such that $\mathbf{x}^*_k > 10^{-n}$. To see this, assume to the contrary and one can see that it is not possible to satisfy the unit simplex constraint. Choose

k as the column in 5.9. Then for any $i \in M$, we have the new column

$$a_{ij} - \min_k a_{kj} + \max_{(i,k)} a_{ik} + \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}_j}\Big|_{x_j=10^{-n}},$$

now for any row $r \in M$, the new matrix element at column k is

$$a_{rk} + \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}_k}\Big|_{\mathbf{x}_k=\mathbf{x}_k^*}.$$

Since the partial derivative function is decreasing and $\mathbf{x}_k^* > 10^{-n}$, we have that

$$\frac{\partial g(\mathbf{x})}{\partial \mathbf{x}_j}\Big|_{x_j=10^{-n}} > \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}_k}\Big|_{\mathbf{x}_k=\mathbf{x}_k^*},$$

and it should hold that

$$-\min_k a_{kj} + \max_{(i,k)} a_{ik} \geq a_{rk} - a_{ij},$$

because $\max_{(i,k)} a_{ik} \geq a_{rk}$ for any (r,k) and $\min_k a_{kj} \leq a_{ij}$ for any $i \in M$. Therefore we obtain

$$a_{ij} - \min_k a_{kj} + \max_{(i,k)} a_{ik} + \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}_j}\Big|_{x_j=10^{-n}} > a_{rk} + \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}_k}\Big|_{\mathbf{x}_k=\mathbf{x}_k^*},$$

for any $i, r \in M$. So we obtain

$$\min_{i\in I}\{a_{ij} - \min_k a_{kj} + \max_{(i,k)} a_{ik} + \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}_j}\Big|_{x_j=10^{-n}}\} > \max_{r\in I}\{a_{rk} + \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}_k}\Big|_{\mathbf{x}_k=\mathbf{x}_k^*}\}.$$

So we can apply 5.3, meaning that the optimum solution to the convex subproblem at iteration $l+1$ must have $\mathbf{x}_j^{l+1} = 0$. □

5.1 particularly holds for any vector values that are zero. This essentially means that in 6, once a variable is set to zero, then it should remain zero until the very end of the algorithm. This is very important as this means that the supports actually become smaller over time.

Corollary 5.1. *Let $\mathbf{x}^l$ denote the solution at the lth iteration of 6, for any $l \in \mathbb{N}$. Then the set of supports is a monotone sequence. That is, we have that*

$$I_1(\mathbf{x}^0) \supseteq I_1(\mathbf{x}^1) \supseteq \ldots \supseteq I_1(\mathbf{x}^l) \supseteq \ldots \tag{5.10}$$

In particular, the sequence of cardinalities is a non increasing sequence. Meaning that

$$\|\mathbf{x}^0\|_0 \geq \|\mathbf{x}^1\|_0 \geq \ldots \|\mathbf{x}^l\|_0 \geq \ldots$$

Proof. The corollary follows from Theorem 5.1, as for any j, $\mathbf{x}_j^l = 0 \implies \mathbf{x}_j^{l+1} = 0$ □

Using 5.1, it is possible to modify 6 to terminate significantly faster. Since the supports are monotone, one can reduce the size of the convex problem at each iteration. The modified version is presented in 8.

Algorithm 8 Modified Iterative DCA (MIDCA) with $\mathcal{G}$

Initialize $\mathbf{x}_0 \in \Delta$, $\beta \in \mathbb{R}$, sparsity level $k < n$, and tolerance T
while $\|\mathbf{x}^{l+1} - \mathbf{x}^l\| \geq T$ **and** $\|\mathbf{x}^l\|_0 > k$ **do**
    Solve the following subproblem:

$$\mathbf{x} \in \arg \min_{\mathbf{x} \in \Delta^{I_1(\mathbf{x}^l)}} \max_i \mathbf{a}_i^{I_1(\mathbf{x}^l)\top} \mathbf{x}^{I_1(\mathbf{x}^l)} + \beta g(\mathbf{x}^{I_1(\mathbf{x}^l)}) + \beta \mathcal{G}(\mathbf{x}_l^{I_1(\mathbf{x}^l)})^\top (\mathbf{x}^{I_1(\mathbf{x}^l)} - \mathbf{x}_l^{I_1(\mathbf{x}^l)})$$

    Construct $\mathbf{x}^{l+1}$ from $\mathbf{x}$, by placing its elements in the correct support.
end while
Let $\mathbf{x}_l^*$ and $\mathbf{x}_{l+1}^*$ be the outputs of this loop.
Obtain $S \subseteq I$ be the indices of the k largest entries of $\mathbf{x}_l^*$. Solve $\mathcal{P}(S)$ and obtain $\mathbf{x}_S^*$.
if $k > |I_1(\mathbf{x}_{l+1})|$ **then**
    Obtain $T \subseteq I$ with the following procedure: Find $k - |I_1(\mathbf{x}_{l+1})|$-many largest elements of $\mathbf{x}_l^*$ that are not in $I_1(\mathbf{x})$. Let I' be this set. Then set $T := I_1(\mathbf{x}) \cup I'$. Solve $\mathcal{P}(T)$ and obtain $\mathbf{x}_T^*$
else
    Set $f(\mathbf{x}_T^*) = +\infty$
end if
Output $\mathbf{x} \in \arg\min\{f(\mathbf{x}_S^*), f(\mathbf{x}_T^*)\}$

Since it solves subproblems rather than large-sized problems, Algorithm 8 is faster than Algorithm 6, yielding the same solutions when terminated. Even though both these algorithms still rely on the β penalty coefficient value, it is possible to find a suitable β after conducting a few trial steps with Algorithm 7. In the Numerical Experiments Chapter, the algorithms are tested using examples from game theory in addition to the real-world examples for binary classification.

Chapter 6

Numerical Experiments

In this chapter, various numerical experiments are conducted to test algorithms developed throughout this thesis. The first set of experiments compares the newly proposed CPFP to well-studied penalties in the literature. Parameters of CPFP are chosen to be (α, γ) pairs of $(0.8, 0.4)$, $(0.6, 0.3)$ and $(0.5, 0.5)$. Choosing $(\alpha, \gamma) = (0.6, 0.3)$ or $(0.8, 0.4)$ is based on the observations in Section 5.3. The selection $(\alpha, \gamma) = (0.5, 0.5)$ is arbitrary, and it is to see the performance of CPFP in different parameter values. Parameter values for other penalties are chosen arbitrarily.

Two main algorithms are proposed to solve $\mathcal{P}$: Algorithm 3 and Algorithm 8. For large-sized examples, recovering the correct support and, hence, the original optimum solution to $\mathcal{P}$ is impossible with known optimization solvers. Therefore, the main benchmark throughout this chapter is the upper bounds of the MILP formulation $\mathcal{P}_{\mathbf{MILP}}$. For large (m, n) values, $\mathcal{P}_{\mathbf{MILP}}$ mostly fails to prove optimality but yields promising upper bounds. As an example, consider a randomly generated game matrix $\mathbf{A}$ sized $m = 2000$ and $n = 1000$. In Figure 6.1, it can be seen that the solver fails to reduce the optimality gap below 98% even after working for 6 hours. In this example, the final upper bound is 2.0097, and the final lower bound is 0.0386.

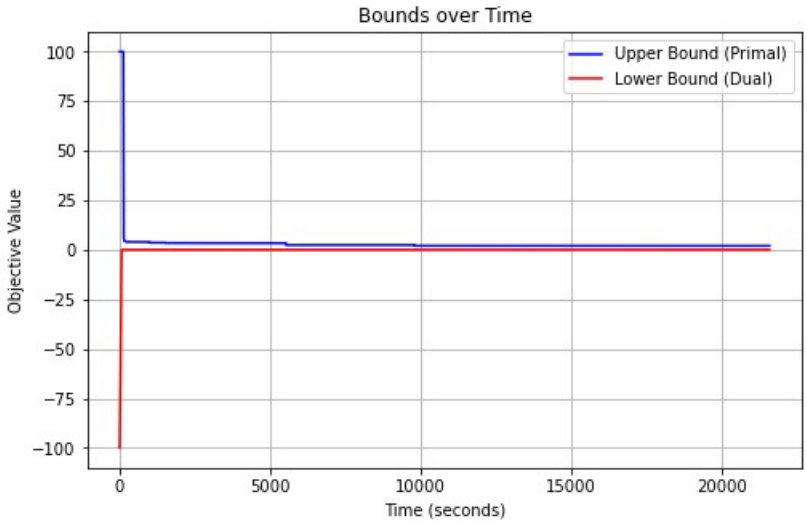

Figure 6.1: Upper and Lower Bounds of $\mathcal{P}$, $\mathbf{A} \in \mathbb{R}^{2000\times1000}$

The methodology of numerical experiments is straightforward. To test both algorithms, each is run until termination, and $\mathcal{P}_{\mathbf{MILP}}$ is run for the same time. An observation from the numerical experiments is that $\mathcal{P}_{\mathbf{MILP}}$ most of the time yields a better solution in the given time frame when initialized with the procedure in Algorithm 4. This is probably due to the fact that the initial solutions found by the solver are generally loose. This can also be verified from the example in Figure 6.1. The final results are compared using various parameters and data for each problem.

First, the results are given for a predetermined number of general game matrices $\mathbf{A}$, as well as for different values of matrix sizes m, n and sparsity level k. In Algorithm 3 and $\mathcal{P}_{\mathbf{MILP}}$, both problems are initialized with the procedure given in 4 based on the observations.

Lastly, MMSCCP defined in Section 1.2 is tested with Algorithm 8, together with the parameter selection procedure in Algorithm 7. Due to its nature, well-known open-source datasets for binary classification can be used for MMSCCP. It is important to highlight that this problem requires many one-level decision stumps to make the classification. Therefore, simple decision stumps are created for every dataset based on the available predictors. The first set of experiments

aims not to classify data correctly but to test the speed and accuracy of Algorithm 8. This chapter concludes with computing margin maximizing solutions for several datasets, trained with the 80% of the data and tested with the rest.

The experiments in this chapter are run with the processor Intel®Core™ i5-8265U CPU @ 1.60GHz and 8GB of RAM.

6.1 Review of the Existing Concave Surrogates

In this section, the performance of the following concave surrogates is assessed under Algorithm 8: Minimax concave penalty (MCP), Concave L_p-norm with $p \in (0, 1)$, $-L_2$ regularization, SCAD and Capped L_1 norm. Penalties that are not concave, such as inverse maximum by [38] are not evaluated. Note that the Piecewise Quadratic penalty by [36] is equivalent to the Negative L_2 penalty over the simplex. Hence, it is not explicitly tested.

The MCP is defined as follows.

$$\mathrm{MCP}_{\theta,\gamma}(x_i) = \begin{cases} \theta x_i - \frac{x_i^2}{2\gamma}, & 0 \leq x_i \leq \gamma\theta \\ \frac{1}{2}\gamma\theta^2, & x_i > \gamma\theta. \end{cases} \tag{6.1}$$

For MCP, test parameters θ and γ are chosen as $0.05, 0.1, 0.3$ and $2, 3$, respectively. γ selection is mostly based on the tests in the original paper [32]. Parameter θ is chosen arbitrarily in (0.1). However, one should choose γ and θ so that $\gamma\theta \leq 1$, because the MCP is then equivalent to

$$\sum_{i=1}^{n} \theta x_i - \frac{x_i^2}{2\gamma} \implies \theta - \sum_{i=1}^{n} \frac{x_i^2}{2\gamma} \implies \theta - \frac{\|\mathbf{x}\|_2^2}{2\gamma}, \tag{6.2}$$

making it analogous to the $-L_2$ penalty over the unit simplex.

Capped L_1 penalty is defined as follows:

$$\mathrm{CappedL1}_{\tau}(x_i) = \min\{x_i, \tau\}. \tag{6.3}$$

It should be noted that there are other versions of this penalty, such as

$$\min\{\theta x_i, \tau\}. \tag{6.4}$$

This version changes the slope of the function when $x_i < \tau$. Indeed, when one computes the derivatives for the DCA, Capped L_1 penalizes the columns with small ($\leq \tau$), thus promoting sparsity.

L_p and $-L_2$ penalties are similar in behavior to CPFP. Those are defined as

$$\|x\|_p^p = \sum_{i=1}^{n} x_i^p, \tag{6.5}$$

where $p \in (0, 1)$ and

$$-\|x\|_2^2 = -\sum_{i=1}^{n} x_i^2. \tag{6.6}$$

Parameter selection for L_p penalty is straightforward, the interval $(0, 1)$ is divided equally and the experiments are conducted for $p = 0.2, 0.4, 0.6, 0.8$.

Lastly, the SCAD is defined as in [34], and the parameter α is chosen to be 3.7 according to the experiments therein.

$$\text{SCAD}_\theta(x_i) = \begin{cases} \theta x_i, & 0 \leq x_i \leq \theta \\ \frac{-x_i^2 + 2a\theta x_i - \theta^2}{2(a-1)}, & \theta < x_i \leq a\theta \\ \frac{(a+1)\theta^2}{2}, & x_i > a\theta, \end{cases} \tag{6.7}$$

The parameter θ determines the slope of this penalty near zero. On the other hand, it determines the threshold values to switch between functions. The values for θ are chosen to be $0.05, 0.1, 0.3$ arbitrarily.

For any parameter, the gradient is computed regularly. Moreover, if the function is nonsmooth at several points, the gradient is assigned to be any element from the subgradients. Note that all of the functions above are concave over Δ. Hence, their multiplicative inverses are subdifferentiable, and it is always possible to generate a subgradient.

The experiments are run for 39 different (m, n, k) triplets. A summary of the results can be found in Table 6.1, and all detailed experiment results can be seen in Appendix B. It can be observed that CPFP and L_p norm penalties yielded the best results in every experiment conducted. For this selection of parameters,

MCP, SCAD, and Capped L_1 penalties usually yield the poorest result. On the other hand, it can be seen that under correct γ selections, CPFP can outperform L_p norm when $\alpha = p$. CPFP seems to be a promising penalty, considering the fact that it can be seen as a generalization of L_p penalties. In most samples, CPFP values are very close to the results in L_p cases. Therefore, choosing the incorrect γ parameter is not harshly penalized in general. The experiments in the following sections are based on the performances of CPFP parameters.

6.2 Finding Sparse Strategies for Two-player Games

For game theory experiments, 5 random matrices for 12 different sizes for NBF and 20 random game matrices for each of 40 different sizes and sparsity values; hence, a total of 800 random matrices are created for MIDCA. Every random matrix $\mathbf{A}$ generated has entries between $a_{lb} = -100$ and $a_{ub} = 100$, and the bounds are chosen arbitrarily. For each problem, parameters (m, n, k) represent the number of rows, columns (features) and the sparsity level, respectively.

6.2.1 Performance of NBFS

The NBF Search algorithm, as in Algorithm 3, is the first approach to solving $\mathcal{P}$. This greedy-type algorithm requires selecting two indices to swap supports. In this version of the algorithm, swapped indices are chosen arbitrarily. This selection can be based on the gradient information at each iteration for differentiable functions. For example, one can choose the leaving index by choosing the index with the largest gradient value among $I_1(\mathbf{x})$ and the entering index with the smallest gradient values among the indices of $I_0(\mathbf{x})$. Although this procedure does not guarantee finding the optimum solution, it provides a sensible way of choosing indices rather than random guessing. Obviously, the gradient selection procedure does not apply to nonsmooth optimization problems. To come up

(m,n,k)	Best Penalties
(60,100,20)	(0.8,0.4), (0.6,0.3),$\mathbf{L_{0.6}}$, $\mathbf{L_{0.8}}$
(60,170,34)	(0.8,0.4), $\mathbf{L_{0.8}}$, $MCP(0.05, 2)$, $MCP(0.1, 2)$,$MCP(0.3, 3)$
(60,240,48)	All penalties yield the same result in this example
(160,100,20)	(0.6,0.3), $\mathbf{L_{0.6}}$
(160,170,34)	(0.6,0.3), $\mathbf{L_{0.6}}$
(160,240,48)	$\mathbf{L_{0.8}}$
(260,100,20)	$\mathbf{L_{0.6}}$
(260,170,34)	$\mathbf{L_{0.6}}$
(260,240,48)	(0.8,0.4)
(360,100,20)	(0.6,0.3), $\mathbf{L_{0.6}}$
(360,170,34)	(0.6,0.3)
(360,240,48)	(0.8,0.4)
(460,100,20)	$\mathbf{L_{0.4}}$
(460,170,34)	(0.6,0.3)
(460,240,48)	(0.6,0.3)
(600,100,14)	(0.6,0.3)
(600,300,42)	$\mathbf{L_{0.6}}$
(600,500,71)	$\mathbf{L_{0.8}}$
(600,700,100)	(0.8,0.4)
(600,900,128)	(0.8,0.4)
(600,1100,157)	$\mathbf{L_{0.8}}$
(700,100,14)	$\mathbf{L_{0.2}}$
(700,300,42)	$\mathbf{L_{0.6}}$
(700,500,71)	(0.8,0.4), $\mathbf{L_{0.8}}$
(700,700,100)	$\mathbf{L_{0.8}}$
(700,900,128)	(0.8,0.4)
(700,1100,157)	$\mathbf{L_{0.8}}$
(800,100,14)	$\mathbf{L_{0.2}}$
(800,300,42)	(0.6,0.3), $\mathbf{L_{0.6}}$
(800,500,71)	(0.6,0.3)
(800,700,100)	$\mathbf{L_{0.8}}$
(800,900,128)	(0.8,0.4)
(800,1100,157)	(0.8,0.4)
(900,100,14)	$\mathbf{L_{0.4}}$
(900,300,42)	(0.6,0.3)
(900,500,71)	$\mathbf{L_{0.8}}$
(900,700,100)	$\mathbf{L_{0.8}}$
(900,900,128)	(0.8,0.4)
(900,1100,157)	$\mathbf{L_{0.8}}$

Table 6.1: Best Penalties for Each Experiment

with a similar heuristic way of choosing leaving and entering variables, one can compute

$$\mathbf{d} \in \arg\min_{\|\mathbf{y}\| \leq 1} \max_{i \in D(\mathbf{x})} \mathbf{a}_i^\intercal \mathbf{y}.$$

Then at each iteration one can choose

$$x_{\text{enter}} \in \arg\min_{i \in I_0(\mathbf{x})} \mathbf{d}_i,$$

and

$$x_{\text{leave}} \in \arg\max_{i \in I_1(\mathbf{x})} \mathbf{d}_i.$$

The vector $\mathbf{d}$ replicates the original gradient, which finds the minimizer vector among the directional derivatives as in Lemma 4.6. The ties in the gradient selection procedure can be broken arbitrarily. NBFS can be costly for large-scale instances, requiring checking $k(n-k)$ neighbors at each iteration to prove the NBF condition. Algorithm 3 can be modified to terminate after reaching an iteration limit L. This way, it terminates in a sub-NBF point but in a significantly shorter amount of time. In the NBF experiments, $L = 1000$ was chosen.

The results of NBFS experiments can be found in Table 6.2. For each row, the average time represents the average runtime of 5 samples, and the average relative improvement is the performance of NBF compared to the MILP upper bounds, on average. The last column indicates the percentage of samples that yield better NBF solutions than MILP.

Even though there are exceptions, one can see that the NBF Algorithm performs comparable to the MILP Upper bounds on the instances with a large number of features. As an example, in the instance with $(800, 520, 112)$, the NBFS performed better in a majority of the samples, and on average improves 3%. The number of instances where the NBFS was better in solutions have an increasing trend as m, n increase. So one could expect NBF to perform slightly better in larger samples, although in these instances the results were comparable. On the other hand, solving the MILP is a better option for relatively smaller values of m, n.

m	n	k	#	Avg. Time	Avg. Impr %	Better %
600	120	26	5	86.08	-4.294	20
600	220	47	5	237.25	-1.886	40
600	320	69	5	228.46	0.14	60
600	420	91	5	285.83	-5.135	0
600	520	112	5	248.38	-1.217	40
800	120	26	5	232.68	0.767	60
800	220	47	5	324.07	-1.409	20
800	320	69	5	352.07	-0.881	40
800	420	91	5	359.06	-1.761	40
800	520	112	5	442.50	0.396	60
1000	120	26	5	284.52	-1.186	20
1000	220	47	5	321.38	-4.672	20

Table 6.2: NBFS vs MILP for different Game Matrices

6.2.2 Performance of MIDCA

The same experiments are run with the MIDCA (Algorithm 8) using the same methodology as the NBFS example. We choose $\beta = 10$ for the MIDCA algorithm, and one can improve our solutions by choosing a more suitable β experimentally. As shown in Table 6.3 and Appendix C.1, the MIDCA achieves better feasible solutions in a significantly shorter time than NBFS and MILP solutions. MIDCA performs better than the MILP solution, even under a relatively small number of features, and achieves an average improvement rate of about 15% on average. In fact, one can run the MIDCA and NBFS algorithms consecutively to seek an even better solution. However, we have observed experimentally that most of the outputs in MIDCA are also NBF points.

Solving relatively small problems (typically $n < 70$) with $\mathcal{P}_{MILP}$ and examples with a larger amount of features with the MIDCA seems to be the best options for solving $\mathcal{P}$. Even though the neighborhood search algorithm scales well compared to the MILP, we suggest using MIDCA in most of the cases, and then using NBFS to search for better neighboring solutions.

m	n	k	#	Avg. Time (s)	Avg. Impr.%	Better %
260	112	24	20	3.66	14.76	100
460	132	28	20	8.66	14.673	100
360	132	28	20	5.78	14.48	100
460	112	24	20	8.90	13.989	100
560	132	28	20	10.54	13.753	100
560	112	24	20	11.26	13.691	100
360	112	24	20	6.60	13.033	100
660	132	28	20	15.59	12.471	100
660	112	24	20	15.03	11.728	100
260	92	19	20	4.44	11.2	90
360	92	19	20	6.21	10.456	100
460	92	19	20	8.13	7.486	85
660	92	19	20	13.85	6.756	95
160	72	15	20	2.00	4.833	60
560	92	19	20	10.95	4.783	75
260	72	15	20	4.12	3.536	60
660	72	15	20	12.08	2.549	65
360	72	15	20	5.78	0.211	50
460	72	15	20	7.45	-0.481	45
560	72	15	20	9.32	-1.739	30
160	52	11	20	1.70	-3.125	35
660	52	11	20	9.15	-4.327	25
360	52	11	20	4.75	-4.354	30
460	52	11	20	6.24	-4.574	20
260	52	11	20	3.60	-5.144	40
560	52	11	20	8.76	-7.795	10
560	12	2	20	0.93	-8.114	0
660	12	2	20	1.15	-8.394	0
360	12	2	20	0.60	-9.371	0
460	12	2	20	0.77	-10.611	0
260	12	2	20	0.39	-14.58	0
560	32	6	20	4.82	-14.821	10
160	12	2	20	0.25	-15.303	0
660	32	6	20	5.79	-16.623	0
260	32	6	20	1.80	-16.789	5
460	32	6	20	3.72	-17.116	5
360	32	6	20	2.93	-18.186	0
160	32	6	20	0.95	-24.852	0
60	12	2	20	0.10	-30.432	0

Table 6.3: MIDCA vs MILP for different Game Matrices ($\beta = 10$,$\alpha = \gamma = 0.5$)

6.3 Margin Maximizing Solutions for Sparse Composite Classifiers via MIDCA

Recall that the MMSCCP is formulated as follows:

$$\begin{aligned} \text{maximize} \quad & \min_{i \in D} \sum_{j \in N} \alpha_j y_i h_j(x_i) \\ \text{subject to} \quad & \alpha \in \Delta^n \\ & \|\alpha\|_0 \leq k. \end{aligned} \tag{6.8}$$

as per the construction in Section 1.2, set

$$z_{ij} = -y_i h_j(x_i),$$

and the resulting problem is

$$\begin{aligned} \text{minimize} \quad & \max_{i \in D} \sum_{j \in N} \alpha_j z_{ij} \\ \text{subject to} \quad & \alpha \in \Delta^n \\ & \|\alpha\|_0 \leq k, \end{aligned} \tag{6.9}$$

which is identical to $\mathcal{P}$. In this case, a critical difference is that the error matrix values are in $\{-1, 1\}$. Since it was observed that MIDCA as in Algorithm 8 is in general superior to NBFS in almost all examples, the experiments in this section use MIDCA with CPFP.

A binary $\{-1, 1\}$ error matrix Z is required for the boosting problem. Datasets available online mostly do not have binary predictors but rather continuous and categorical variables for prediction. In the context of boosting, classifiers are required instead of the original variable values. Therefore, the first step is to create binary classifiers h_j given a dataset. One could consider a plethora of methods to convert the data into a classification matrix. In this set of experiments, the aim is to compute classifiers with large margins in a short amount of time rather than the classification performance. Therefore, the thresholding technique is used to construct the "correct" and "incorrect" labels. For every predictor, the methodology is to simply separate data points through the median and label with $\{-1, 1\}$.

A downside of thresholding is that it does not capture extreme values of a predictor, as these are treated equally depending on the value compared to the median. As an example, consider the Colon Cancer dataset by [65]. This experiment aims to predict colon cancer in patients by looking at gene intensity values. Separating intensity values through the median could yield poorly performing classifiers in practice because a gene of a patient being extremely intense compared to other specimens could be a reason to trigger cancer. To create accurate boosting classifiers, one can create better classifiers/decision stumps with any method of choice, and creating the classifiers is out of the scope of this thesis.

Experiments with MIDCA and CPFP are conducted in a similar fashion to the previous chapters. The results can be seen in Table 6.4. In this table, DC and MILP columns shows the minimum margins generated by each method, respectively. Note that this problem is a maximization problem and one seeks larger minimum margins. The Time column represents the runtime it takes for MIDCA with β search to terminate, and the last column shows the β value upon termination. The experiments use $\alpha = 0.5, \gamma = 0.5$ as CPFP parameters. Note that in the *Scene* dataset, the target class was "Urban", and in the others the target of classification is evident from the sources.

Dataset	**m**	**n**	**k**	**DC**	$\mathcal{P}_{\mathbf{MILP}}$	**Time**	β
Bioresp. [66]	3751	1776	40	1	1	258.28	0.0000
InternetAds [67]	3279	1558	40	1	1	194.967	0.0000
Madelon [68]	2600	500	40	-0.26266	-0.2661	194.44	0.3125
Scene [69]	2407	299	40	-0.35865	-0.35809	55.55	0.1563
OVA Colon [65]	1545	10935	40	-0.13177	-0.16045	846.48	0.1563
Random	1000	1000	40	-0.17064	-0.2134	143.18	0.1563
Kidney [65]	546	10935	40	0.29372	0.29034	232.50	0.0391
Prostate [70]	102	12533	40	0.35473	0.35489	47.15	0.0195

Table 6.4: Comparing MIDCA with CPFP in Open Source Datasets

It can be seen that the results are similar to Section 6.2, and the MIDCA with CPFP outperforms MILP upper bounds in most of the examples. The performance of MIDCA is more evident when both the data size m and the number of available decision stumps n is large.

Since the emphasis was to *find* the sparse margin maximizing classifiers, so far no optimization have been done for the purposes of error minimization. The minimax formulation provides robustness and might improve generalization performance of the model. However, small error rates might not be expected directly. Also, the generalization performance should depend on the way the decision stumps are created. As a result, no experiments on training and testing data for classification was run directly, and it was left open for future work combining sparse error minimization and margin maximization.

Chapter 7

Conclusion

This thesis extended and developed three optimality conditions and proposed two main algorithms to solve the CCMOP. The motivation for solving CCMOP comes from the applications in computing sparse strategies in bimatrix games and finding sparse margin maximizing composite classifiers in the boosting problem. The problem is unique in that it has a nonsmooth objective over the sparse unit simplex. Studying nonsmooth problems theoretically is harder in practice because convergence results mostly fail due to the lack of gradient. Moreover, the special structure of the unit simplex allows one to consider penalties with different geometric properties.

7.1 Optimality Conditions and Algorithms

Optimality conditions to $\mathcal{P}$ are firstly expressed as an extension of the BF and CW conditions in the literature for smooth optimization problems. In the spirit of the previous work, Basic Feasible points represent the optimum solutions of each k-subset of the variables and are the most natural conditions for optimality. This condition is reduced to a subgradient condition for the minimax function

over the unit simplex, and it can be checked by solving a linear system of equations. Coordinatewise (CW) optimality is inherited from the literature to search for better local solutions. Due to the nonsmooth nature, CW points can be poor in performance. The Neighbor Basic Feasibility condition is developed as an extension of both BF and CW points. Starting from an initial feasible point, the NBF Search algorithm is proposed to find candidate NBF points to the original optimum solution. The thesis assesses the performance of greedy algorithms to find points that satisfy the developed optimal conditions. The NBF Algorithm performs poorly in small game matrices but improves significantly for large samples, outperforming the Mixed-Integer formulation upper bounds in several cases. Indeed, NBFS can perform significantly better if a more efficient method of index selection is developed.

7.2 CPFP and DCA Approach

Since the study of optimality conditions is limited, the connections between the CCMOP and CRMOP were established through the local behavior of the minimax function. In particular, it was shown that the CRMOP local optimum points coincide with the BF points of the CCMOP. Therefore, solving for CRMOP local optimum points is equivalent to solving for CCMOP. A drawback of this reformulation is that the CRMOP is a discontinuous problem. A novel surrogate penalty, CPFP, is developed to overcome this irregularity and promote sparsity over unit simplex. CPFP is applicable to all optimization problems over the unit simplex, although its performance should be verified for the general case. Analysis of penalty parameters shows that for most instances, $\alpha \in [0.6, 0.8]$ performs the best, where the γ parameter can be adjusted for better solution quality. A Difference of Convex approach is utilized in solving $\mathcal{P}$ with CPFP. It was shown that DCA, in this case, has a special interpretation of altering the columns of the game matrix. For MIDCA, the algorithm was tested with random game matrices and real binary classification data. MIDCA performs much better in numerical experiments than MILP formulation, even in relatively small samples, $n > 90$ in particular. A search procedure is designed for cases where the β parameter is

unknown from the context.

7.3 Future Directions and Extensions

The main goal of this thesis was to develop techniques to find qualified feasible solutions to $\mathcal{P}$. On the other hand, it was observed that the MILP formulation fails to generate tight lower bounds when solved with state-of-the-art solvers. The study of lower bounds to $\mathcal{P}$ was not attempted but could be interesting in future work. Optimality conditions in Chapter 3 can be extended significantly. The problem $\mathcal{P}$ is not approached from the perspective of variational analysis, and the study of optimality conditions is limited to the BF, CW, and NBF conditions. Although the special properties of the minimax function were used in the proofs, the connections between $\mathcal{P}$ and $\mathcal{P}_{\mathbf{Reg}}$ should be similar in any convex and continuous function. Lastly, the sparse boosting approach is only developed for binary classification. The results should extend to the n-ary case, and could be extended to a mixed setting including error minimization and margin maximization.

In summary, this thesis lays a foundation for further study of sparsity constrained nonsmooth optimization problems on constrained domains. Future research can build on the theoretical properties, penalty development, and algorithms presented here.

Data Availability.

The data used in this thesis are publicly available from the references indicated for each data set.

Bibliography

[1] M. Sion, "On general minimax theorems.," *Pacific Journal of Mathematics*, vol. 8, no. 1, pp. 171 – 176, 1958.

[2] D.-Z. Du and P. M. Pardalos, *Minimax and applications*. Springer, 2013.

[3] J. von Neumann, "Zur theorie der gesellschaftsspiele," *Mathematische Annalen*, vol. 100, p. 295–320, Dec. 1928.

[4] R. J. Lipton, E. Markakis, and A. Mehta, "Playing large games using simple strategies," in *Proceedings of the 4th ACM Conference on Electronic Commerce*, EC '03, (New York, NY, USA), p. 36–41, Association for Computing Machinery, 2003.

[5] R. J. Lipton and N. E. Young, "Simple strategies for large zero-sum games with applications to complexity theory," in *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '94, (New York, NY, USA), p. 734–740, Association for Computing Machinery, 1994.

[6] X. Yuan, P. Li, and T. Zhang, "Gradient hard thresholding pursuit for sparsity-constrained optimization," in *Proceedings of the 31st International Conference on Machine Learning* (E. P. Xing and T. Jebara, eds.), vol. 32(2) of *Proceedings of Machine Learning Research*, (Bejing, China), pp. 127–135, PMLR, 22–24 Jun 2014.

[7] D. Bertsimas, J. Pauphilet, and B. Van Parys, "Sparse classification: a scalable discrete optimization perspective," *Machine Learning*, vol. 110, p. 3177–3209, Nov. 2021.

[8] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, p. 119–139, Aug. 1997.

[9] A. J. Ferreira and M. A. T. Figueiredo, "Boosting algorithms: A review of methods, theory, and applications," in *Ensemble Machine Learning: Methods and Applications* (C. Zhang and Y. Ma, eds.), pp. 35–85, New York, NY: Springer New York, 2012.

[10] A. M. Tillmann, D. Bienstock, A. Lodi, and A. Schwartz, "Cardinality minimization, constraints, and regularization: A survey," *SIAM Review*, vol. 66, no. 3, pp. 403–477, 2024.

[11] M. Gaudioso, G. Giallombardo, and G. Miglionico, "Sparse optimization via vector k-norm and dc programming with an application to feature selection for support vector machines," *Computational Optimization and Applications*, vol. 86, p. 745–766, July 2023.

[12] M. S. Daskin, *Network and discrete location: Models, algorithms, and applications*. Wiley, 1995.

[13] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. USA: W. H. Freeman & Co., 1990.

[14] O. P. Burdakov, C. Kanzow, and A. Schwartz, "Mathematical programs with cardinality constraints: Reformulation by complementarity-type conditions and a regularization method," *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 397–425, 2016.

[15] D. Bienstock, "Computational study of a family of mixed-integer quadratic programming problems," *Math. Program.*, vol. 74, pp. 121–140, Aug. 1996.

[16] D. Bertsimas and R. Shioda, "Algorithm for cardinality-constrained quadratic optimization," *Comput. Optim. Appl.*, vol. 43, pp. 1–22, May 2009.

[17] A. Beck and Y. C. Eldar, "Sparsity constrained nonlinear optimization: Optimality conditions and algorithms," *SIAM Journal on Optimization*, vol. 23, no. 3, pp. 1480–1509, 2013.

[18] A. Beck and N. Hallak, "On the minimization over sparse symmetric sets: Projections, optimality conditions, and algorithms," *Mathematics of Operations Research*, vol. 41, no. 1, pp. 196–223, 2016.

[19] S. Lämmel and V. Shikhman, "On nondegenerate m-stationary points for sparsity constrained nonlinear optimization," *Journal of Global Optimization*, vol. 82, p. 219–242, Aug. 2021.

[20] M. Xue and L. Pang, "A strong sequential optimality condition for cardinality-constrained optimization problems," *Numerical Algorithms*, vol. 92, p. 1875–1904, Aug. 2022.

[21] Z. Xiao and J. J. Ye, "Optimality conditions and constraint qualifications for cardinality constrained optimization problems," *Numerical Algebra, Control and Optimization*, vol. 14, no. 3, p. 614–635, 2024.

[22] C. Kanzow, A. B. Raharja, and A. Schwartz, "Sequential optimality conditions for cardinality-constrained optimization problems with applications," *Computational Optimization and Applications*, vol. 80, p. 185–211, July 2021.

[23] J. Y. Zhang, R. Khanna, A. Kyrillidis, and O. O. Koyejo, "Learning sparse distributions using iterative hard thresholding," in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[24] J. Pan and M. Yan, "Efficient sparse probability measures recovery via bregman gradient," *Journal of Scientific Computing*, vol. 102, Jan. 2025.

[25] M. Nikolova, "Description of the minimizers of least squares regularized with l_0-norm. uniqueness of the global minimizer," *SIAM Journal on Imaging Sciences*, vol. 6, no. 2, pp. 904–937, 2013.

[26] M. Nikolova, "Relationship between the optimal solutions of least squares regularized with l_0-norm and constrained by k-sparsity," *Applied and Computational Harmonic Analysis*, vol. 41, no. 1, pp. 237–265, 2016. Sparse

Representations with Applications in Imaging Science, Data Analysis and Beyond.

[27] N. Zhang and Q. Li, "On optimal solutions of the constrained l_0 regularization and its penalty problem," *Inverse Problems*, vol. 33, p. 025010, Jan. 2017.

[28] D. Ge, X. Jiang, and Y. Ye, "A note on the complexity of l p minimization," *Mathematical Programming*, vol. 129, p. 285–299, June 2011.

[29] Y. Chen, D. Ge, M. Wang, Z. Wang, Y. Ye, and H. Yin, "Strong NP-hardness for sparse optimization with concave penalty functions," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 740–747, PMLR, 06–11 Aug 2017.

[30] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 58, p. 267–288, Jan. 1996.

[31] T. Zhang, "Analysis of multi-stage convex relaxation for sparse regularization," *Journal of Machine Learning Research*, vol. 11, no. 35, pp. 1081–1107, 2010.

[32] C.-H. Zhang, "Nearly unbiased variable selection under minimax concave penalty," *The Annals of Statistics*, vol. 38, no. 2, pp. 894–942, 2010.

[33] S. Foucart and M.-J. Lai, "Sparsest solutions of underdetermined linear systems via l_q-minimization for $0 < q \leq 1$," *Applied and Computational Harmonic Analysis*, vol. 26, no. 3, pp. 395–407, 2009.

[34] J. Fan and R. Li, "Variable selection via nonconcave penalized likelihood and its oracle properties," *Journal of the American Statistical Association*, vol. 96, no. 456, pp. 1348–1360, 2001.

[35] E. Soubies, L. Blanc-Féraud, and G. Aubert, "A continuous exact l_0 penalty (cel0) for least squares regularized problem," *SIAM Journal on Imaging Sciences*, vol. 8, no. 3, pp. 1607–1639, 2015.

[36] Q. Li, Y. Bai, C. Yu, and Y. Yuan, "A new piecewise quadratic approximation approach for ℓ_0 norm minimization problem," *Science China Mathematics*, vol. 62, no. 1, pp. 185–204, 2019.

[37] E. Soubies, L. Blanc-Féraud, and G. Aubert, "A Unified View of Exact Continuous Penalties for L_2-L_0 Minimization," *SIAM Journal on Optimization*, vol. 27, no. 3, 2017.

[38] M. Pilanci, L. El Ghaoui, and V. Chandrasekaran, "Recovery of sparse probability measures via convex programming," in *Advances in Neural Information Processing Systems*, vol. 25, pp. 2420–2428, 2012.

[39] P. Li, S. S. Rangapuram, and M. Slawski, "Methods for sparse and low-rank recovery under simplex constraints," *Statistica Sinica*, 2020.

[40] T. Blumensath and M. E. Davies, "Iterative thresholding for sparse approximations," *Journal of Fourier Analysis and Applications*, vol. 14, p. 629–654, Sept. 2008.

[41] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[42] W. Bian and X. Chen, "A smoothing proximal gradient algorithm for nonsmooth convex regression with cardinality penalty," *SIAM Journal on Numerical Analysis*, vol. 58, p. 858–883, Jan. 2020.

[43] H. Le Thi, T. Pham Dinh, H. Le, and X. Vo, "Dc approximation approaches for sparse optimization," *European Journal of Operational Research*, vol. 244, p. 26–46, July 2015.

[44] T. Lipp and S. Boyd, "Variations and extension of the convex–concave procedure," *Optimization and Engineering*, vol. 17, p. 263–287, Nov. 2015.

[45] W. Li, W. Bian, and K.-C. Toh, "Difference-of-convex algorithms for a class of sparse group ℓ_0 regularized optimization problems," *SIAM Journal on Optimization*, vol. 32, p. 1614–1641, July 2022.

[46] I. Althöfer, "On sparse approximations to randomized strategies and convex combinations," *Linear Algebra and its Applications*, vol. 199, pp. 339–355, 1994. Special Issue Honoring Ingram Olkin.

[47] S. Afiouni, J. Černý, C. K. Ling, and C. Kroer, "Title of the paper," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39(13), pp. 13502–13509, Apr. 2025.

[48] A. J. Ferreira and M. A. Figueiredo, "Boosting algorithms: A review of methods, theory, and applications," in *Ensemble machine learning: Methods and applications* (C. Zhang and Y. Ma, eds.), Springer, 2012.

[49] P. Beja-Battais, "Overview of AdaBoost : Reconciling its views to better understand its dynamics." working paper or preprint, Oct. 2023.

[50] S. Rosset, J. Zhu, and T. Hastie, "Margin maximizing loss functions," in *Advances in Neural Information Processing Systems* (S. Thrun, L. Saul, and B. Schölkopf, eds.), vol. 16, MIT Press, 2003.

[51] G. Rätsch and M. K. Warmuth, "Efficient margin maximizing with boosting," *Journal of Machine Learning Research*, vol. 6, no. 71, pp. 2131–2152, 2005.

[52] A. J. Grove and D. Schuurmans, "Boosting in the limit: maximizing the margin of learned ensembles," in *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI '98/IAAI '98, (USA), p. 692–699, American Association for Artificial Intelligence, 1998.

[53] Y. T. Xi, Z. J. Xiang, P. J. Ramadge, and R. E. Schapire, "Speed and sparsity of regularized boosting," *International Conference on Artificial Intelligence and Statistics*, p. 615–622, Apr. 2009.

[54] P. Bühlmann and B. Yu, "Sparse boosting," *Journal of Machine Learning Research*, vol. 7, no. 36, pp. 1001–1024, 2006.

[55] T. Zhang and B. Yu, "Boosting with early stopping: Convergence and consistency," *The Annals of Statistics*, vol. 33, Aug. 2005.

[56] Z. Xu, G. Huang, K. Q. Weinberger, and A. X. Zheng, "Gradient boosted feature selection," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, (New York, NY, USA), p. 522–531, Association for Computing Machinery, 2014.

[57] S. Z. Gilani, F. Shafait, and A. Mian, "Gradient based efficient feature selection," in *IEEE Winter Conference on Applications of Computer Vision*, pp. 191–197, 2014.

[58] A. Beck, *Introduction to Nonlinear Optimization: Theory, Algorithms, and Applications with Python and MATLAB, Second Edition*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2023.

[59] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms I*. Springer Berlin Heidelberg, 1993.

[60] Y. Nesterov, *Nonsmooth Convex Optimization*, pp. 139–240. Cham: Springer International Publishing, 2018.

[61] R. T. Rockafellar and R. J. B. Wets, *Max and Min*, pp. 1–37. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998.

[62] J. R. Munkres, *Topology*. Upper Saddle River, NJ: Prentice Hall, 2nd ed., 2000. See Section 27, Exercise 27.2(b) for the result.

[63] P. D. Tao and L. T. H. An, "A d.c. optimization algorithm for solving the trust-region subproblem," *SIAM Journal on Optimization*, vol. 8, no. 2, pp. 476–505, 1998.

[64] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[65] G. Stiglic and P. Kokol, "Stability of ranked gene lists in large microarray analysis studies," *Journal of Biomedicine and Biotechnology*, vol. 2010, p. 1–9, 2010.

[66] B. Hamner, D. C. Thompson, and Jorg, "Predicting a biological response." `https://www.kaggle.com/competitions/bioresponse`, 2012. Kaggle competition.

[67] N. Kushmerick, "Internet Advertisements." UCI Machine Learning Repository, 1999. DOI: https://doi.org/10.24432/C5V011.

[68] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror, "Result analysis of the nips 2003 feature selection challenge," in *Advances in Neural Information Processing Systems* (L. Saul, Y. Weiss, and L. Bottou, eds.), vol. 17, MIT Press, 2004.

[69] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, "Mulan: A java library for multi-label learning," *Journal of Machine Learning Research*, vol. 12, no. 71, pp. 2411–2414, 2011.

[70] D. Singh, P. G. Febbo, K. Ross, D. G. Jackson, J. Manola, C. Ladd, P. Tamayo, A. A. Renshaw, A. V. D'Amico, J. P. Richie, E. S. Lander, M. Loda, P. W. Kantoff, T. R. Golub, and W. R. Sellers, "Gene expression correlates of clinical prostate cancer behavior," *Cancer Cell*, vol. 1, p. 203–209, Mar. 2002.

Appendix A

Results of Parameter Analysis

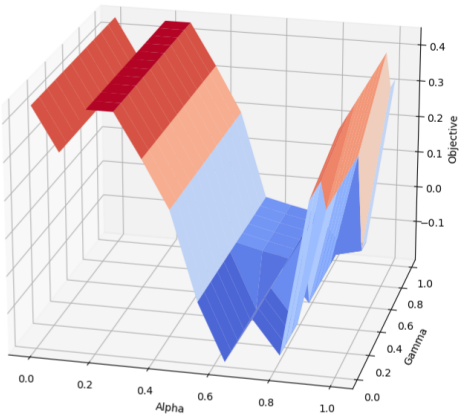

Figure A.1: Analysis of CPFP α and γ Parameters, 100×100 Sample

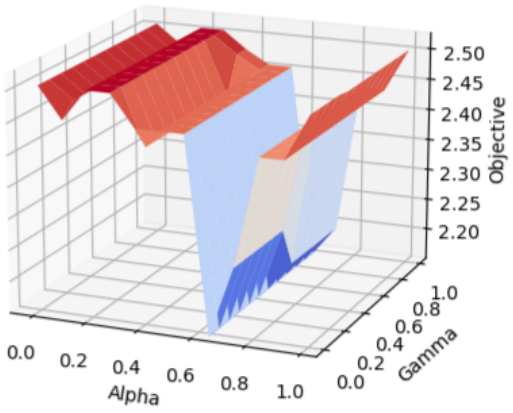

Figure A.2: Analysis of CPFP α and γ Parameters, 150×150 Sample

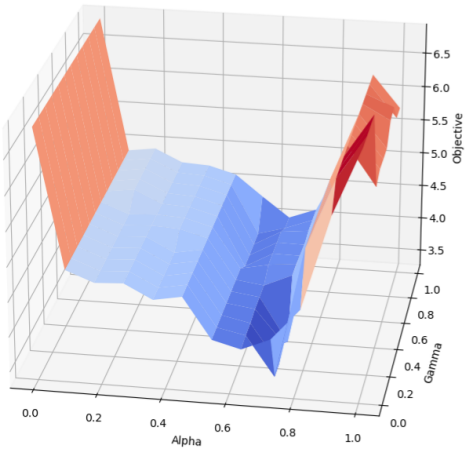

Figure A.3: Analysis of CPFP α and γ Parameters, 200×200 Sample

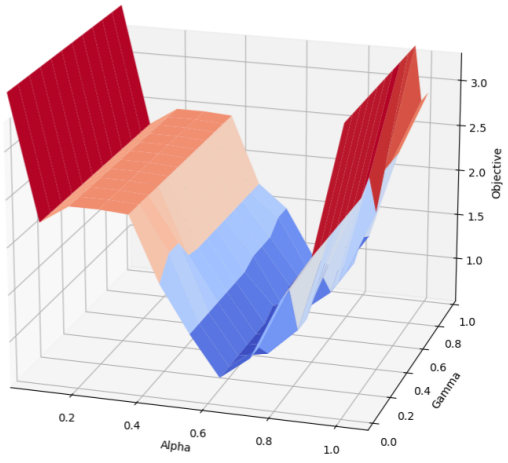

Figure A.4: Analysis of CPFP α and γ Parameters, 150×200 Sample

Appendix B

Penalty Comparison

(m,n,k)	**(0.8,0.4)**	**(0.6,0.3)**	**(0.5,0.5)**
(60,100,20)	-1.368	-1.368	-0.807
(60,170,34)	-6.141	-6.049	-6.049
(60,240,48)	-8.916	-8.916	-8.916
(160,100,20)	10.557	8.425	8.524
(160,170,34)	1.685	1.22	2.145
(160,240,48)	-1.329	-0.737	-0.642
(260,100,20)	12.96	13.902	12.317
(260,170,34)	5.605	5.576	5.67
(260,240,48)	1.66	2.342	2.642
(360,100,20)	17.733	14.616	15.445
(360,170,34)	8.721	7.372	7.708
(360,240,48)	4.475	4.698	4.921
(460,100,20)	20.721	18.015	17.447
(460,170,34)	9.895	9.577	9.684
(460,240,48)	6.701	6.075	6.691
(600,100,14)	29.22	24.329	26.875
(600,300,42)	8.944	8.024	8.576
(600,500,71)	3.38	3.742	4.298
(600,700,100)	1.279	1.932	2.08
(600,900,128)	0.383	0.925	0.938
(600,1100,157)	-0.329	0.151	0.255
(700,100,14)	29.456	30.185	26.105
(700,300,42)	9.911	8.6	9.512
(700,500,71)	4.434	4.667	5.401
(700,700,100)	2.036	2.448	2.822
(700,900,128)	1.064	1.701	1.725
(700,1100,157)	-0.147	0.619	0.661
(800,100,14)	33.765	33.37	30.907
(800,300,42)	10.561	9.19	9.661
(800,500,71)	5.912	5.509	6.266
(800,700,100)	2.631	3.725	4.098
(800,900,128)	1.247	2.004	2.306
(800,1100,157)	0.442	1.017	1.018
(900,100,14)	32.619	32.132	28.538
(900,300,42)	12.486	10.117	10.491
(900,500,71)	6.365	6.465	7.049
(900,700,100)	2.8	3.437	4.274
(900,900,128)	1.93	2.487	3.335
(900,1100,157)	1.024	1.725	1.942

Table B.1: Results for Different α, γ Values

(m,n,k)	$L_{0.2}$	$L_{0.4}$	$L_{0.6}$	$L_{0.6}$	$-L_2$
(60,100,20)	-0.684	-0.807	-1.368	-1.368	-0.807
(60,170,34)	-6.049	-6.049	-6.049	-6.141	-6.112
(60,240,48)	-8.916	-8.916	-8.916	-8.916	-8.916
(160,100,20)	9.92	9.691	8.425	10.074	9.661
(160,170,34)	2.933	2.071	1.22	2.118	3.059
(160,240,48)	-0.642	-0.642	-0.801	-1.407	-0.642
(260,100,20)	13.673	12.285	11.907	13.821	14.969
(260,170,34)	7.921	6.057	5.467	5.648	8.239
(260,240,48)	2.731	2.731	2.342	1.917	2.562
(360,100,20)	15.854	15.274	14.616	19.332	18.994
(360,170,34)	9.009	8.266	7.488	8.591	9.851
(360,240,48)	5.388	4.965	4.74	4.751	5.858
(460,100,20)	16.894	16.189	17.031	20.721	21.179
(460,170,34)	11.852	10.736	9.774	10.73	12.583
(460,240,48)	7.447	6.87	6.198	6.696	7.699
(600,100,14)	25.269	24.851	26.552	29.22	32.615
(600,300,42)	10.223	9.112	7.972	9.057	11.299
(600,500,71)	4.821	4.793	3.742	3.354	4.977
(600,700,100)	2.328	2.328	1.932	1.284	2.246
(600,900,128)	0.938	0.938	0.901	0.422	1.036
(600,1100,157)	0.255	0.255	0.175	-0.356	0.099
(700,100,14)	24.173	25.715	29.652	29.456	30.489
(700,300,42)	11.058	9.965	8.467	10.352	11.589
(700,500,71)	6.398	6.233	4.667	4.434	6.547
(700,700,100)	3.222	3.204	2.448	2.019	3.07
(700,900,128)	1.725	1.725	1.701	1.089	1.657
(700,1100,157)	0.661	0.661	0.619	-0.155	0.568
(800,100,14)	28.371	29.638	28.675	35.987	34.413
(800,300,42)	11.584	10.659	9.19	10.38	12.053
(800,500,71)	7.256	7.172	5.513	5.882	7.182
(800,700,100)	4.203	4.203	3.725	2.619	4.096
(800,900,128)	2.306	2.306	2.004	1.288	2.126
(800,1100,157)	1.018	1.018	1.005	0.445	1.047
(900,100,14)	30.222	28.287	32.689	33.125	29.426
(900,300,42)	11.787	11.527	10.123	12.157	12.772
(900,500,71)	7.523	7.087	6.392	5.962	7.457
(900,700,100)	4.4	4.4	3.437	2.688	4.525
(900,900,128)	3.349	3.349	2.531	1.949	3.306
(900,1100,157)	1.942	1.942	1.726	1.007	1.739

Table B.2: Results for L_p, $p \in (0, 1)$ and $-L_2$

m	**n**	**k**	**(m,n,k)**	$\tau = 0.1$	$\tau = 0.3$	$\tau = 0.5$
60	100	20	(60,100,20)	-0.724	-0.684	-0.684
60	170	34	(60,170,34)	-6.049	-6.049	-6.049
60	240	48	(60,240,48)	-8.916	-8.916	-8.916
160	100	20	(160,100,20)	11.661	11.661	11.661
160	170	34	(160,170,34)	2.933	2.933	2.933
160	240	48	(160,240,48)	-0.642	-0.642	-0.642
260	100	20	(260,100,20)	15.415	15.415	15.415
260	170	34	(260,170,34)	7.921	7.921	7.921
260	240	48	(260,240,48)	2.731	2.731	2.731
360	100	20	(360,100,20)	18.914	18.914	18.914
360	170	34	(360,170,34)	9.682	9.682	9.682
360	240	48	(360,240,48)	5.388	5.388	5.388
460	100	20	(460,100,20)	21.298	21.298	21.298
460	170	34	(460,170,34)	12.859	12.859	12.859
460	240	48	(460,240,48)	7.787	7.787	7.787
600	100	14	(600,100,14)	28.202	28.202	28.202
600	300	42	(600,300,42)	11.325	11.325	11.325
600	500	71	(600,500,71)	4.815	4.815	4.815
600	700	100	(600,700,100)	2.251	2.251	2.251
600	900	128	(600,900,128)	1.032	1.032	1.032
600	1100	157	(600,1100,157)	0.045	0.045	0.045
700	100	14	(700,100,14)	34.502	34.502	34.502
700	300	42	(700,300,42)	12.433	12.433	12.433
700	500	71	(700,500,71)	5.447	5.447	5.447
700	700	100	(700,700,100)	3.862	3.862	3.862
700	900	128	(700,900,128)	2.001	2.001	2.001
700	1100	157	(700,1100,157)	0.773	0.773	0.773
800	100	14	(800,100,14)	31.914	31.914	31.914
800	300	42	(800,300,42)	12.992	12.992	12.992
800	500	71	(800,500,71)	6.931	6.931	6.931
800	700	100	(800,700,100)	3.87	3.87	3.87
800	900	128	(800,900,128)	2.718	2.718	2.718
800	1100	157	(800,1100,157)	1.328	1.328	1.328
900	100	14	(900,100,14)	35.812	35.812	35.812
900	300	42	(900,300,42)	12.595	12.595	12.595
900	500	71	(900,500,71)	7.512	7.512	7.512
900	700	100	(900,700,100)	5.609	5.609	5.609
900	900	128	(900,900,128)	3.05	3.05	3.05
900	1100	157	(900,1100,157)	1.484	1.484	1.484

Table B.3: Results for Capped L_1

m	**n**	**k**	**(m,n,k)**	$\gamma = 0.05$	$\gamma = 0.1$	$\gamma = 0.3$
60	100	20	(60,100,20)	-0.684	-0.684	-0.684
60	170	34	(60,170,34)	-6.049	-6.049	-6.049
60	240	48	(60,240,48)	-8.916	-8.916	-8.916
160	100	20	(160,100,20)	11.661	11.661	11.661
160	170	34	(160,170,34)	2.933	2.933	2.933
160	240	48	(160,240,48)	-0.642	-0.642	-0.642
260	100	20	(260,100,20)	15.415	15.415	15.415
260	170	34	(260,170,34)	7.921	7.921	7.921
260	240	48	(260,240,48)	2.731	2.731	2.731
360	100	20	(360,100,20)	18.914	18.914	18.914
360	170	34	(360,170,34)	9.682	9.682	9.682
360	240	48	(360,240,48)	5.388	5.388	5.388
460	100	20	(460,100,20)	21.298	21.298	21.298
460	170	34	(460,170,34)	12.859	12.859	12.859
460	240	48	(460,240,48)	7.787	7.787	7.787
600	100	14	(600,100,14)	28.202	28.202	28.202
600	300	42	(600,300,42)	11.325	11.325	11.325
600	500	71	(600,500,71)	4.815	4.815	4.815
600	700	100	(600,700,100)	2.251	2.251	2.251
600	900	128	(600,900,128)	1.032	1.032	1.032
600	1100	157	(600,1100,157)	0.045	0.045	0.045
700	100	14	(700,100,14)	34.502	34.502	34.502
700	300	42	(700,300,42)	12.433	12.433	12.433
700	500	71	(700,500,71)	5.447	5.447	5.447
700	700	100	(700,700,100)	3.862	3.862	3.862
700	900	128	(700,900,128)	2.001	2.001	2.001
700	1100	157	(700,1100,157)	0.773	0.773	0.773
800	100	14	(800,100,14)	31.914	31.914	31.914
800	300	42	(800,300,42)	12.992	12.992	12.992
800	500	71	(800,500,71)	6.931	6.931	6.931
800	700	100	(800,700,100)	3.87	3.87	3.87
800	900	128	(800,900,128)	2.718	2.718	2.718
800	1100	157	(800,1100,157)	1.328	1.328	1.328
900	100	14	(900,100,14)	35.812	35.812	35.812
900	300	42	(900,300,42)	12.595	12.595	12.595
900	500	71	(900,500,71)	7.512	7.512	7.512
900	700	100	(900,700,100)	5.609	5.609	5.609
900	900	128	(900,900,128)	3.05	3.05	3.05
900	1100	157	(900,1100,157)	1.484	1.484	1.484

Table B.4: Results for SCAD, $\alpha = 3.7$

m	**n**	**k**	**(m,n,k)**	$(0.05, 2)$	$(0.1, 2)$	$(0.3, 3)$
60	100	20	(60,100,20)	-0.807	-0.807	-0.807
60	170	34	(60,170,34)	-6.141	-6.141	-6.141
60	240	48	(60,240,48)	-8.916	-8.916	-8.916
160	100	20	(160,100,20)	11.363	11.363	11.661
160	170	34	(160,170,34)	3.059	3.059	3.059
160	240	48	(160,240,48)	-0.642	-0.642	-0.642
260	100	20	(260,100,20)	15.097	15.097	15.097
260	170	34	(260,170,34)	7.743	7.743	7.743
260	240	48	(260,240,48)	2.562	2.562	2.731
360	100	20	(360,100,20)	18.914	18.914	18.914
360	170	34	(360,170,34)	9.534	9.534	9.851
360	240	48	(360,240,48)	5.506	5.506	5.926
460	100	20	(460,100,20)	21.298	21.298	21.298
460	170	34	(460,170,34)	12.859	12.859	12.859
460	240	48	(460,240,48)	7.658	7.658	7.658
600	100	14	(600,100,14)	32.712	32.712	32.712
600	300	42	(600,300,42)	11.325	11.325	11.325
600	500	71	(600,500,71)	4.815	4.815	4.815
600	700	100	(600,700,100)	2.27	2.27	2.251
600	900	128	(600,900,128)	0.988	0.988	1.032
600	1100	157	(600,1100,157)	-0.017	-0.017	0.055
700	100	14	(700,100,14)	33.156	33.156	33.156
700	300	42	(700,300,42)	11.81	11.81	11.758
700	500	71	(700,500,71)	5.452	5.452	5.452
700	700	100	(700,700,100)	3.884	3.884	3.884
700	900	128	(700,900,128)	1.92	1.92	1.92
700	1100	157	(700,1100,157)	0.842	0.842	0.826
800	100	14	(800,100,14)	32.889	32.889	32.889
800	300	42	(800,300,42)	12.979	12.979	12.979
800	500	71	(800,500,71)	6.931	6.931	6.931
800	700	100	(800,700,100)	3.907	3.907	3.907
800	900	128	(800,900,128)	2.67	2.67	2.718
800	1100	157	(800,1100,157)	1.256	1.256	1.228
900	100	14	(900,100,14)	35.812	35.812	35.812
900	300	42	(900,300,42)	12.595	12.595	12.595
900	500	71	(900,500,71)	7.466	7.466	7.512
900	700	100	(900,700,100)	5.415	5.415	5.415
900	900	128	(900,900,128)	3.044	3.044	3.05
900	1100	157	(900,1100,157)	1.569	1.569	1.568

Table B.5: Results for MCP

Appendix C

MIDCA vs MILP Comparison

m	n	k	#	Avg. Time (s)	Avg. Impr.%	Better %
460	132	28	20	13.050	14.223	100
560	132	28	20	16.471	12.747	100
260	112	24	20	5.361	10.471	95
360	132	28	20	8.457	13.785	95
660	132	28	20	22.514	9.036	95
460	112	24	20	12.058	9.004	90
560	112	24	20	15.046	9.878	90
360	112	24	20	8.764	9.823	85
660	112	24	20	19.367	8.234	85
460	92	19	20	10.575	5.312	80
260	92	19	20	4.914	4.490	65
360	92	19	20	7.612	4.104	65
560	92	19	20	13.613	1.286	65
660	92	19	20	17.415	1.489	60
260	72	15	20	4.325	-0.899	50
160	72	15	20	2.008	-0.883	45
460	72	15	20	9.153	-4.484	35
360	72	15	20	6.525	-4.909	25
560	72	15	20	10.523	-5.428	25
160	52	11	20	1.590	-11.508	20
360	52	11	20	4.890	-11.757	20
460	52	11	20	6.819	-8.657	20
660	72	15	20	13.541	-5.117	20
560	52	11	20	8.269	-10.273	10
660	52	11	20	10.437	-10.162	10
560	32	6	20	4.453	-15.518	5
60	12	2	20	0.086	-32.969	0
160	12	2	20	0.290	-15.180	0
160	32	6	20	0.957	-23.968	0
260	12	2	20	0.352	-14.565	0
260	32	6	20	1.603	-19.822	0
260	52	11	20	3.174	-13.747	0
360	12	2	20	0.564	-9.371	0
360	32	6	20	2.509	-19.514	0
460	12	2	20	0.693	-10.693	0
460	32	6	20	3.335	-19.493	0
560	12	2	20	0.884	-7.791	0
660	12	2	20	0.969	-8.479	0
660	32	6	20	5.058	-16.714	0

Table C.1: MIDCA Results with CPFP, $\alpha = 0.6$,$\gamma = 0.3$