SOLVING ELLIPSOIDAL INCLUSION AND OPTIMAL EXPERIMENTAL DESIGN PROBLEMS: THEORY AND ALGORITHMS.

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Selin Damla Ahipaşaoğlu

August 2009

SOLVING ELLIPSOIDAL INCLUSION AND OPTIMAL EXPERIMENTAL DESIGN PROBLEMS:
THEORY AND ALGORITHMS.

Selin Damla Ahipaşaoğlu, Ph.D.
Cornell University 2009

This thesis is concerned with the development and analysis of Frank-Wolfe type algorithms for two problems, namely the ellipsoidal inclusion problem of optimization and the optimal experimental design problem of statistics. These two problems are closely related to each other and can be solved simultaneously as discussed in Chapter 1 of this thesis.

Chapter 1 introduces the problems in parametric forms. The weak and strong duality relations between them are established and the optimality criteria are derived. Based on this discussion, we define ϵ-primal feasible and ϵ-approximate optimal solutions: these solutions do not necessarily satisfy the optimality criteria but the violation is controlled by the error parameter ϵ and can be arbitrarily small.

Chapter 2 deals with the most well-known special case of the optimal experimental design problems: the D-optimal design problem and its dual, the Minimum-Volume Enclosing Ellipsoid (MVEE) problem. Chapter 3 focuses on another special case, the A-optimal design problem. In Chapter 4 we focus on a generalization of the optimal experimental design problem in which a subset but not all of parameters is being estimated. We focus on the following two problems: the D_k-optimal design problem and the A_k-optimal design problem, generalizations of the D-optimal and the A-optimal design problems, re-

spectively. In each chapter, we develop first-order algorithms for the respective problem and discuss their global and local convergence properties. We present various initializations and provide some computational results which confirm the attractive features of the first-order methods discussed.

Chapter 5 investigates possible combinatorial extensions of the previous problems. Special attention is given to the problem of finding the Minimum-Volume Ellipsoid (MVE) estimator of a data set, in which a subset of a certain size is selected so that the minimum-volume ellipsoid enclosing these points has the smallest volume. We discuss how the algorithms in Chapter 2 can be adapted in order to attack this problem. Many efficient heuristics and a branch-and-bound algorithm are developed.

BIOGRAPHICAL SKETCH

Selin Damla Ahipaşaoğlu was born in Erzurum, a city in Eastern Turkey. Nevertheless, she has spent almost all of her childhood and early youth, in Ankara, the capital of Turkey.

She was at Bilkent University studying Industrial Engineering between 1998 and 2002, studying hard for the never-ending projects and exams but still enjoying heart-lifting nights with gals at the dorm. She liked it so much that she has stayed for another two years and got an M.S. degree from the same department. Rumor has it, some of her many days in the library doing literature review on location theory were actually unintentionally spent on fantasy fiction novels without her former advisors knowledge.

Bored of Ankara at last, she has moved to gorges(!) Ithaca in 2004 in order to pursue a Ph.D. in mathematical programming at Cornell University. She has been working with Michael J. Todd since then and have learned a lot about optimization, computer science, and statistics. After five years of hard work, she is in love with ellipsoids and Cholesky factors and she is well aware of the geekiness of this statement. Since she is still not bored of being a student, she will move to Princeton University, for a postdoctoral position after her graduation. She is looking forward to lay on the beach and do nothing but read fiction and play computer games in between though.

to my dear family...

ACKNOWLEDGEMENTS

I would like to express my gratitute to many great people who have had a very positive influence in my life. Without them, I would be less than what I am now.

I would like to thank my advisor Michael J. Todd, not only for his excellent academic guidance but also for being such a great person. He was always very supportive, kind, and humorous. I have given him very hard time with my interesting use of *the* English language and punctuation many times, but he was very patient. I believe that I have learned a lot from him and I am very proud to be his student. I also would like to thank other members of my committee, Adrian Lewis and Charles van Loan for reading my thesis. I would like to thank Adrian Lewis and David Shmoys for their support and attention during my studies. They are excellent teachers in the classroom and it is always a pleasure to listen to them. I hope to be such a good teacher myself one day.

I am indebted to my family. My mother, Saniye, was a great role model for all my life. A strong woman and an excellent teacher. She taught me the joy of learning. My aunts, Leyla and Şaziye, and grandma have given their deepest love to me, I know that it was too hard for them to let me study in a place so far away from home. I am thankful to their love and support.

I was lucky enough to know many great people at Cornell. I would like to thank all Ph.D. students at Cornell ORIE for making our department such a friendly and lively environment. Of course Alex, Bikram, Dennis, and Spyros have a special place in my heart since we have been through many difficult times together, especially in the first couple of months in Ithaca. I would like to thank Tuncay Alparslan. He is one of the finest people that I have known in many senses. He shared his passion for research, science fiction, cooking, and drinking with me. Turan Birol and I have shared endless hours talking about

literature and many weird subjects during long winter nights, these turned out to be very warm nights indeed. I am honored to be friends with Ilham and Mahir Çipil, I felt their support all the time. Ilham has been my dear *muse*. Yasin Alan made me laugh many times, especially whenever I felt like crying and giving up. Last but never the least, I would like to thank Burak Ulgut, who has been an excellent friend for 11 years. He has been my housemate, teammate, drinking buddy, therapist, best friend, and brother. We drove some 10000 miles and flew a lot more than that together. My life would be very dull without the presence of these people, and I hope that they will continue to enlightening and spicing up my life.

Spyros Schismenos is the single person with whom I have shared it all. Together we worked, learned a lot, made stupid math jokes, travelled, cooked, got drunk, made fun of others, been made fun of, got deeply depressed, and got stupidly happy. I am actually thankful to Cornell for accepting both of us in 2004. We figured out *what Ithacas really mean* after all.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

'Do not disturb my circles!', Archimedes, 212 B.C.

In this chapter, we introduce the optimal experimental design problem in a parametric form. We start with the statistical motivation, discuss common properties of the optimal design problems, and formulate the problem rigorously. We continue by introducing the ellipsoidal inclusion problem in a general form. We show that this problem is a generalization of some of the well-known inclusion problems such as the Minimum-Volume Enclosing Ellipsoid problem and the Minimum Enclosing Ball problem. Weak and strong duality relations between these statistical and geometric problems are established and optimality criteria are derived. Based on this discussion, we define ϵ-primal feasible and ϵ-approximate optimal solutions: these solutions do not necessarily satisfy the optimality criteria but the violation is controlled by the error parameter ϵ and can be arbitrarily small.

The problems covered in this chapter have applications in optimization, statistics, data mining, outlier detection, machine learning, image processing, etc. These applications usually lead to very large problems. Devising algorithms that can attack very large instances of these two problems and their extensions is the goal of this thesis.

1.1 Experimental Design

Conducting experiments and analyzing the outcomes of these experiments using statistical models is a common task for many scientists. Usually a relationship such as

$$y = h(t, \theta) \tag{1.1.1}$$

is assumed, where y is the observed outcome of the experiment and h is a function of the variables t and θ. The function h is a real-valued function of two distinct arguments: The first argument t is a vector representing the experimental conditions. These conditions are chosen or *designed* freely by the experimenter. The second argument θ is a vector holding the *parameters of the system*. The experimenter has no control over these values, i.e., the parameter vector is determined by the nature of the experiment. It is the goal of the experimenter to learn as much as possible about the function h and the parameter vector θ so that she can make accurate predictions about the responses of the system.

It is obvious that the model function h plays an important role. The so-called generalized linear model

$$y = x(t)^T \theta, \tag{1.1.2}$$

is used extensively as a model function. Despite its simplicity, it proves to be useful for many applications. In the generalized linear model, $x(t) = [x_1(t); \ldots; x_n(t)]$ and $\theta = [\theta_1; \ldots; \theta_n]$ are column vectors in n-dimensional Euclidean space. Any nonlinear relations between the experimental conditions can be incorporated into the model using the function $x(\cdot)$, but the model is required to be linear with respect to the system parameters. In this thesis, we will suppress the dependency of the vector $x(t)$ on the actual experimental conditions t

and work with a model function such as

$$y = x^T\theta, \tag{1.1.3}$$

in which the vector x will be referred to as the regression or design vector.

The model function in (1.1.3) is deterministic in nature. Unfortunately, almost all experiments carried out are subject to some random error. Measurement errors due to devices or humans and incorrect model assumptions are typical sources of experimental errors. It is commonly assumed that random errors are additive in nature so that (1.1.3) is replaced by

$$y = x^T\theta + \epsilon, \tag{1.1.4}$$

where the response y and the error ϵ are random variables with an unknown distribution function. One important assumption is that the error in one run of the experiment is independent of the error in another run of the experiment. Without this assumption, the model becomes too complicated.

In order to use this model we need some information about the nature of the error distribution. If the only goal of model building were parameter estimation, moderate assumptions on the first and second moments of the distribution would suffice. Nevertheless, since these models are further used in hypothesis testing and interval estimation, it is quite common to assume that the random variable ϵ is normally distributed with mean 0 and unknown variance σ^2.

Let $\mathcal{X} = \{x_1, \ldots, x_m\} \subset \mathbb{R}^n$ be the set of regression vectors and X denote a matrix of order $n \times m$ whose columns consist of these vectors. Assume henceforth that the x_i's span $\mathbb{R}^n$. If we assume that one experiment is conducted at each of these regression points and the model at hand is a generalized linear model with the normality assumption discussed in the previous paragraph, it is well-known

that the optimal (unbiased) estimator for the parameter vector θ is $(X^TX)^{-1}X^TY$, and has dispersion matrix $\sigma^2(X^TX)^{-1}$. It may be desirable to conduct multiple or zero experiments at some of the regression points, and so we have the following definition:

Definition 1.1.1 *An* experimental design of size N *is given by a finite number of regression points* $x_1, \ldots, x_m$ *in* $\mathbb{R}^n$ *and nonnegative integers* $n_1, \ldots, n_m$ *such that* $\sum_{i=1}^m n_i = N$.

In an experimental design, n_i represents the number of repetitions of the experiment at the regression point x_i. The *dispersion matrix* becomes

$$D = \sigma^2 \left(\sum_{i=1}^m n_i x_i x_i^T \right)^{-1} = \frac{\sigma^2}{N} \left(\sum_{i=1}^m \frac{n_i}{N} x_i x_i^T \right)^{-1}. \tag{1.1.5}$$

Optimal experimental design focuses on finding integers n_i so that the dispersion matrix, which is a measure of the variance (or the error) of the estimator, is minimized in some sense. (Frequently, the regression points can also be chosen from some fixed compact set $\hat{\mathcal{X}}$. Since we are interested in numerical algorithms, we suppose that some large fixed subset $\mathcal{X}$ of $\hat{\mathcal{X}}$ has been preselected, and we are only interested in choosing the n_i's.)

It is easy to see that the dispersion matrix D belongs to the cone of symmetric positive semidefinite matrices, $S\mathbb{R}_+^{n\times n}$. It is reasonable to be interested in the matrices which are minimal with respect to the following (partial) ordering in this set, the Loewner ordering:

Definition 1.1.2 *The Loewner ordering* $\succeq$ *is defined on the subspace of the* $n \times n$ *symmetric matrices by* $A \succeq B \iff A - B \in S\mathbb{R}_+^{n\times n}$.

We also write $S\mathbb{R}^{n\times n}_{++}$ for the cone of symmetric positive definite matrices and write $A \succ B$ to mean $A - B \in S\mathbb{R}^{n\times n}_{++}$. Since the Loewner ordering is antitonic, i.e., $A \succeq B$ implies that $B^{-1} \succeq A^{-1}$, minimizing D is equivalent to maximizing the *information matrix*

$$M = \frac{N}{\sigma^2} \sum_{i=1}^{m} \frac{n_i}{N} x_i x_i^T.$$

When the total number of experiments N is finite, experimental design problems become integer programming problems which are quite hard to attack especially for large m. Hence the case where N tends to infinity is studied instead. In this case we can write $u_i = \frac{n_i}{N}$ and maximizing $M = \sum_{i=1}^{m} u_i x_i x_i^T$ is enough. Note that an experimental design with an infinite sample size N defines a probability distribution which assigns all its weight to a finite number of points. The points with positive weight are the *support points* of the experimental design. One can refer to Chapter 12 in [22] for a valuable discussion on how to come up with an approximate design for a finite sample size once the optimal design for an infinite sample size is found.

When solving experimental design problems, finding a solution which is minimal with respect to the Loewner ordering is necessary. Nevertheless, optimization using this criterion is not straightforward and the set of matrices that satisfy it is usually very large. We prefer real-valued criteria which still preserve Loewner optimality.

Definition 1.1.3 *An* information function *is a function ϕ from the cone of positive semidefinite matrices to the real line,*

$$\phi : S\mathbb{R}^{n\times n}_{+} \to \mathbb{R},$$

which is positively homogeneous, superadditive, nonnegative, nonconstant, and upper

semicontinuous.

It is easy to see that information functions are concave. They order the information matrices according to their informative value. The most common information functions are matrix means.

Definition 1.1.4 *Let $\lambda(C)$ denote the eigenvalues of a matrix C. If C is a positive definite matrix, i.e., $C > 0$, the matrix mean ϕ_p is a function defined as*

$$\phi_p(C) = \begin{cases} \lambda_{\max}(C) & \text{for} \quad p = \infty; \\ \left(\frac{1}{n}\text{Trace}C^p\right)^{1/p} & \text{for} \quad p \neq 0, \pm\infty; \\ (\det C)^{1/n} & \text{for} \quad p = 0; \\ \lambda_{\min}(C) & \text{for} \quad p = -\infty. \end{cases}$$

If C is a singular positive semidefinite matrix, then

$$\phi_p(C) = \begin{cases} \lambda_{\max}(C) & \text{for} \quad p = \infty; \\ \left(\frac{1}{n}\text{Trace}C^p\right)^{1/p} & \text{for} \quad p \neq 0, \infty; \\ 0 & \text{for} \quad p \leq 0. \end{cases}$$

Matrix means satisfy the necessary properties of information functions when $p \leq 1$. Using these functions, the *general optimal design problem* is defined as follows:

$$(\mathcal{D}_p) \qquad \begin{aligned} \max_u \quad g_p(u) &:= \ln \phi_p(XUX^T) \\ e^T u &= 1, \\ u &\geq 0, \end{aligned}$$

where $U := \text{Diag}(u)$. Each value of the parameter p gives rise to a different criterion with different applications. We will study some of the special cases in the following chapters in great detail.

1.2 Ellipsoidal Inclusion Problems

Assume that we have a set of points $\mathcal{X} = \{x_1, \ldots, x_m\} \subset \mathbb{R}^n$, which spans $\mathbb{R}^n$ and is symmetric with respect to the origin. We are interested in approximating (especially enclosing) the convex hull of these points with an ellipsoid. Note that the idea is to approximate the complex structure of the convex hull with a simple geometric object. Boxes, balls, ellipsoids, and cylinders are used in the literature. Ellipsoids are preferred in many applications since they are smooth and flexible, and testing membership in or optimizing a linear function over an ellipsoid is a straightforward task.

The set

$$\mathcal{E}(\bar{x}, H) := \{x \in \mathbb{R}^n : (x - \bar{x})^T H (x - \bar{x}) \leq n\}$$

for $\bar{x} \in \mathbb{R}^n$ and $H \succ 0$ is an ellipsoid in $\mathbb{R}^n$. It is centered at $\bar{x}$ and its shape is defined by H. It can be viewed as a unit ball under an affine map where each point $\tilde{x}$ in the unit ball is mapped to a point $x = \bar{x} + \sqrt{n} L \tilde{x}$ in the ellipsoid where L satisfies $LL^T = H^{-1}$. Geometric properties of the ellipsoid such as its volume, length of its semi-axes, etc., are determined by the shape matrix H. For example, its volume is $\frac{n^{n/2}}{\sqrt{\det H}}$ times that of the unit ball.

The convex hull of a set of finitely many points can be enclosed by an infinite number of ellipsoids. Obviously we are only interested in ellipsoids which are centered at the origin (since $\mathcal{X}$ is symmetric around the origin) and resemble the convex hull in some sense. Although the enclosing ellipsoid which has the minimum volume is a natural choice from both theoretical and practical points of view, as will be discussed in detail in Chapter 2, defining the problem using a more general criterion is quite insightful.

For $q \leq 1$, consider the following problem:

$$
\begin{array}{rrcl}
 & \min_H \quad f_q(H) & := & -\ln \phi_q(H) \\
(\mathcal{P}_q) & x_i^T H x_i & \leq & n,\ i = 1, \ldots, m, \\
 & H & \succ & 0.
\end{array}
$$

For each value of q, this problem finds an ellipsoid which encloses all points in $\mathcal{X}$, is centered at the origin, and has a shape matrix with the largest matrix mean ϕ_q. Each value of the parameter q leads to a different problem with a different geometric interpretation. For example, when $q = 0$, the objective function becomes (a multiple of) $\ln \det(H^{-1})$ and hence $(\mathcal{P}_q)$ is equivalent to the Minimum-Volume Enclosing Ellipsoid problem discussed in Chapter 2. Similarly, for the extreme case of $q = -\infty$, we have $\ln(\lambda_{\min}(H))^{-1}$ as the objective function and hence the problem becomes that of finding the Minimum Enclosing Ball of $\mathcal{X}$. When $q = 1/2$, $(\mathcal{P}_q)$ maximizes the trace of $H^{1/2}$ and leads to a less familiar geometric problem in which we would like to maximize the sum of the inverses of the semi-axes of the enclosing ellipsoid. This problem will be studied in Chapter 3. We will refer to the general problem $(\mathcal{P}_q)$ as the *ellipsoidal inclusion* problem.

1.3 Duality

We now show that the two problems introduced above are closely related to each other.

Lemma 1.3.1 [Weak Duality] *Let p and q be a pair of conjugate numbers in $(-\infty, 1)$, i.e., they satisfy $pq = p + q$. Then we have $f_q(H) \geq g_p(u)$ for any H and u feasible in $(\mathcal{P}_q)$ and $(\mathcal{D}_p)$, respectively.*

Proof: We have

$$\begin{aligned} f_q(H) - g_p(u) &= -\ln \phi_q(H) - \ln \phi_p(XUX^T) \\ &= -\ln\left(\phi_q(H)\phi_p(XUX^T)\right) \\ &\geq -\ln\left(\frac{1}{n} H \bullet XUX^T\right) \\ &\geq -\ln 1 = 0, \end{aligned}$$

where $\bullet$ denotes the trace product of two symmetric matrices, i.e., $A \bullet B = \text{Trace}(AB)$. The first inequality is an application of the Hölder's inequality (on the eigenvalues of the matrices at hand) and a detailed proof can be found in [22]. The second inequality follows from the feasibility of the solutions H and u. Indeed, $\frac{1}{n} H \bullet (XUX^T) = \frac{1}{n} \sum_{i=1}^{m} \left(u_i H \bullet (x_i x_i^T)\right) \leq \frac{1}{n} \sum_{i=1}^{m} \left(u_i (x_i^T H x_i)\right) \leq \frac{n}{n} = 1$. □

Theorem 1.3.1 [Strong Duality] *Let p and q be a pair of conjugate numbers in $(-\infty, 1)$. There exist optimal solutions for problems $(\mathcal{P}_q)$ and $(\mathcal{D}_p)$. Furthermore, the following conditions, together with primal and dual feasibility, are necessary and sufficient for optimality in both $(\mathcal{P}_q)$ and $(\mathcal{D}_p)$:*

a. $H = \frac{n}{\text{Trace}(XUX^T)^p} (XUX^T)^{p-1}$ *and*

b. $x_i^T H x_i = n$ *if* $u_i > 0$.

Proof: Let H be a feasible solution for problem $(\mathcal{P}_q)$. Summing up the linear constraints, we must have $\sum_{i=1}^{m} x_i^T H x_i = H \bullet XX^T \leq nm$. Since $XX^T \succ 0$ and $nm > 0$, $\{H \succeq 0 : H \bullet XX^T \leq nm\}$ is a compact set. Hence the feasible region for problem $(\mathcal{P}_q)$ is also a compact set (since it is the intersection of a compact set with a finite set of halfspaces). Moreover, $H = \epsilon I$ is feasible for $(\mathcal{P}_q)$ for sufficiently small positive ϵ, and we can add the constraint that $f_p(H) \leq f_p(\epsilon I)$

without loss of generality. The objective function is (finite and) continuous on this modified compact feasible region, so an optimal solution exists for problem $(\mathcal{P}_q)$. Existence of an optimal solution for $(\mathcal{P}_q)$ implies the existence of an optimal solution for $(\mathcal{D}_p)$ as will be discussed later.

Sufficiency follows from the previous lemma, since the conditions imply equality in the weak duality inequality. In order to prove necessity, let $\tilde{H}$ be an optimal solution for $(\mathcal{P}_q)$. The KKT conditions must hold for this solution, i.e., there exist nonnegative multipliers $\tilde{u} \in \mathbb{R}^m$ such that the following equalities hold:

$$-\frac{n}{\text{Trace}\tilde{H}^q}\tilde{H}^{q-1} + X\tilde{U}X^T = 0, \tag{1.3.6}$$

$$\tilde{u}_i(n - x_i^T\tilde{H}x_i) = 0, \quad i = 1, \ldots, m. \tag{1.3.7}$$

These equalities imply that $\sum_{i=1}^m \tilde{u}_i = 1$, since

$$\begin{aligned}
\sum_{i=1}^m \tilde{u}_i &= \frac{\sum_{i=1}^m \tilde{u}_i x_i^T \tilde{H} x_i}{n} \\
&= \text{Trace}\left(\frac{\tilde{H}X\tilde{U}X^T}{n}\right) \\
&= \text{Trace}\left(\frac{\tilde{H}\left(\frac{n}{\text{Trace}\tilde{H}^q}\tilde{H}^{q-1}\right)}{n}\right) \\
&= \frac{n\text{Trace}H^q}{n\text{Trace}H^q} = 1,
\end{aligned}$$

and hence $\tilde{u}$ is a feasible solution for $(\mathcal{D}_p)$. Strong duality holds for the solution pair $\tilde{H}$ and $\tilde{u}$, so strong duality holds for any pair of optimal solutions H and u. Conditions (a) and (b) are direct consequences of Equations (1.3.6) and (1.3.7), and hence they are necessary. □

Let $\beta_i(u) := x_i^T(XUX^T)^{p-1}x_i$. The following identity will be used extensively.

$$u^T\beta(u) = \sum_{i=1}^m u_i\beta_i(u)$$

$$
\begin{aligned}
&= \sum_{i=1}^{m} \text{Trace}\left(u_i x_i^T (XUX^T)^{p-1} x_i\right) \\
&= \text{Trace}\left((XUX^T)^{p-1} \sum_{i=1}^{m} u_i x_i x_i^T\right) \\
&= \text{Trace}(XUX^T)^p. \qquad (1.3.8)
\end{aligned}
$$

Using (1.3.8), we can write the necessary and sufficient conditions for u to be optimal in $(\mathcal{D}_q)$ (the optimal H follows from (a)) as

(i) $\beta_i(u) \leq u^T\beta(u)$ for all i, and

(ii) $\beta_i(u) = u^T\beta(u)$ if $u_i > 0$,

which motivates the following definitions.

Definition 1.3.1 *Given a positive ϵ, we call a dual feasible point u an* ϵ-primal feasible solution *if $\beta_i(u) \leq u^T\beta(u)(1+\epsilon)$ for all i, and say that it satisfies the* ϵ-approximate optimality conditions *or it is an* ϵ-approximate optimal solution *if moreover $\beta_i(u) \geq u^T\beta(u)(1-\epsilon)$ whenever $u_i > 0$.*

The following lemma justifies the notation and proves that an ϵ-primal feasible solution for $(\mathcal{D}_p)$ is close to being optimal in a well-defined way.

Lemma 1.3.2 *Let p and q be a pair of conjugate numbers in $(-\infty, 1)$. Given a dual feasible solution u which is ϵ-primal feasible, $H = \frac{n}{(1+\epsilon)\text{Trace}(XUX^T)^p}(XUX^T)^{p-1}$ is feasible in $(\mathcal{P}_q)$ and we have $0 \leq g_p^* - g_p(u) \leq \ln(1+\epsilon)$ where g_p^* is the optimal objective function value of $(\mathcal{D}_p)$.*

Proof: The ϵ-primal feasibility implies that $H = \frac{n}{(1+\epsilon)\text{Trace}(XUX^T)^p}(XUX^T)^{p-1}$ is feasible for the primal problem $(\mathcal{P}_q)$. Let us first assume that $p, q \neq 0$. Then by weak

duality, we have

$$\begin{aligned}
0 &\leq f_q(H) - g_p^* \\
&= -\frac{1}{q}\ln\left(\frac{1}{n}\text{Trace}\left(\frac{n(XUX^T)^{p-1}}{(1+\epsilon)\text{Trace}(XUX^T)^p}\right)^q\right) - g_p^* \\
&= \ln(1+\epsilon) - \frac{1}{q}\ln\left(\frac{n^{q-1}\text{Trace}(XUX^T)^{(p-1)q}}{(\text{Trace}(XUX^T)^p)^q}\right) - g_p^* \\
&= \ln(1+\epsilon) + \ln\left(n^{\frac{1-q}{q}}\left(\text{Trace}(XUX^T)^p\right)^{\frac{q-1}{q}}\right) - g_p^* \\
&\leq \ln(1+\epsilon) + \frac{1}{p}\ln\left(\frac{1}{n}\text{Trace}(XUX^T)^p\right) - g_p^* \\
g_p^* - g_p(u) &\leq \ln(1+\epsilon).
\end{aligned}$$

The case where $p = q = 0$ is very similar and will be proved in Chapter 2. □

Lemma 1.3.3 *$u^0 = \frac{1}{m}(1, 1, \ldots, 1)$ is an (m-1)-primal feasible solution.*

Proof: We have

$$\begin{aligned}
\sum_{i=1}^m \frac{1}{m}\beta_i(u^0) &= (u^0)^T\beta(u^0), \text{ or} \\
\sum_{i=1}^m \beta_i(u^0) &= m(u^0)^T\beta(u^0), \text{ so that} \\
\max_{1\leq i\leq m}\beta_i(u^0) &\leq (1+(m-1))(u^0)^T\beta(u^0),
\end{aligned}$$

and the result follows from the definition of an (m-1)-primal feasible solution. □

So far, we have developed the duality relation between problems $(\mathcal{P}_q)$ and $(\mathcal{D}_p)$ and characterized the optimal solutions of these problems. We have an initial solution for $(\mathcal{D}_p)$ which is somewhat close to optimality and we can assess the quality of the solutions at hand. In other words, we know how to start and end an algorithm for $(\mathcal{D}_p)$ and now we need to figure out how to move from a

given solution to a better one. In the following chapters, we will discuss some of the special cases of these problems in detail and develop algorithms that solve both problems.

1.4 Applications

Before moving into details of solution techniques, we emphasize here that the problems studied in this thesis find applications in various areas. Optimal design problems are useful in linear regression as already discussed. Ellipsoid inclusion problems arise in data analysis and computational geometry (see the references in [31, 18]). They are also used in collision analysis as a subroutine (see Rimon and Boyd [23]) and help in calculating efficient kernels in machine learning (see Shawe-Taylor and Cristianini [27]). The MVE estimator (discussed in Chapter 5) provides a metric which is widely used in outlier detection (see Rousseeuw and Leroy [26], Poston and Tolson [20], and Hawkins [12]). In addition, the inclusion problem also arises as a subproblem in optimization, e.g., for each iteration of the ellipsoid method of Yudin and Nemirovskii [38] and Shor [28] (where a closed-form solution is available) or to initialize Lenstra's integer programming algorithm [19]. Since most of these applications involve large data sets in medium to high dimensions, coming up with techniques which can attack large instances is a natural goal. The following pages describe many positive steps towards accomplishing this goal.

1.5 Notation

We include a summary of the notation that will be used extensively in the following chapters here:

$$
\begin{aligned}
\beta_i(u) &:= x_i^T (XUX^T)^{p-1} x_i; \\
\xi_i(u) &:= x_i^T (XUX^T)^{-1} x_i; \\
\alpha_i(u) &:= x_i^T (XUX^T)^{-2} x_i; \\
\zeta_i(u) &:= z_i^T (ZUZ^T)^{-1} z_i; \\
\omega_i(u) &:= (y_i + Ez_i)^T K^{-1} (y_i + Ez_i); \\
\xi_{ij}(u) &:= x_i^T (XUX^T)^{-1} x_j; \\
\alpha_{ij}(u) &:= x_i^T (XUX^T)^{-2} x_j; \\
\zeta_{ij}(u) &:= z_i^T (ZUZ^T)^{-1} z_j; \\
\omega_{ij}(u) &:= (y_i + Ez_i)^T K^{-1} (y_j + Ez_j); \\
\lambda &:= \frac{\tau}{1-\tau}; \\
\mu &:= \frac{\lambda}{1 + \lambda \zeta_j(u)}; \text{ and} \\
\eta &:= \frac{\lambda}{1 + \lambda \xi_j(u)} = \frac{\mu}{1 + \mu \omega_j(u)}.
\end{aligned}
$$

CHAPTER 2

THE DETERMINANT CRITERION

Suppose we are given a matrix $X = [x_1, x_2, \ldots, x_m] \in \mathbb{R}^{n\times m}$ whose columns, the points $x_1, \ldots, x_m$, span $\mathbb{R}^n$. If H is a symmetric matrix of order n that is positive definite (we write $H \succ 0$), then the volume of the ellipsoid

$$E(0, H) := \{x \in \mathbb{R}^n : x^T H x \leq n\}$$

is $(\det H)^{-1/2}$ times that of a Euclidean ball in $\mathbb{R}^n$ of radius $\sqrt{n}$. Hence, finding a minimum-volume central (i.e., centered at the origin) ellipsoid containing the columns of X amounts to solving

$$\begin{array}{cccl} & \min_{H\succ 0} & f(H) & := & -\ln\det H \\ (\mathcal{P}) & & x_i^T H x_i & \leq & n,\ i = 1, \ldots, m. \end{array}$$

This is called the minimum-volume enclosing ellipsoid (MVEE) problem. It is a special case of the problem $(\mathcal{P}_q)$ in Chapter 1 where $q = 0$.

Khachiyan [16] showed that the seemingly more general problem of finding a not necessarily central ellipsoid of minimum volume containing a finite point set in $\mathbb{R}^n$ reduces to the MVEE problem for a related point set in $\mathbb{R}^{n+1}$, so we henceforth consider only the central case.

Problem $(\mathcal{P})$ is convex, with linear inequality constraints. After some simplification, its Lagrangian dual can be written as

$$\begin{array}{cccl} & \max_u & g(u) & := & \ln\det XUX^T \\ (\mathcal{D}) & & e^T u & = & 1, \\ & & u & \geq & 0, \end{array}$$

where matrix U is a diagonal matrix with the components of u on its diagonal and e is a vector of ones in $\mathbb{R}^m$. Problem $(\mathcal{D})$ is also the statistical problem of finding a D-optimal design measure on the columns of X, that is, one that maximizes the determinant of the Fisher information matrix $E[xx^T]$: see, e.g., Atwood [3, 4], Fedorov [7], John and Draper [14], Kiefer and Wolfowitz [17], Silvey [30], and Wynn [35, 36]. As expected, $(\mathcal{D})$ is a special case of the general optimal experimental design problem $(\mathcal{D}_p)$ discussed in Chapter 1 where $p = 0$. Duality of the problems $(\mathcal{P})$ and $(\mathcal{D})$ can be deduced from Lemma 1.3.1 and Theorem 1.3.1.

In [4], Atwood developed an algorithm for $(\mathcal{D})$ that is a simple modification of those of Fedorov [7] and Wynn [35]. Indeed, we will see in Section 2.1 that Atwood's method is a specialization to $(\mathcal{D})$ of the Frank-Wolfe algorithm [8] with Wolfe's away steps [33] and review the global convergence properties of this algorithm under special initial conditions. We will prove linear convergence of the objective function values for both $(\mathcal{P})$ and $(\mathcal{D})$ using the Wolfe-Atwood method in Section 2.2.

Note that $(\mathcal{D})$ is a convex problem, with feasible region the unit simplex. For such problems, Wolfe [33] sketched the proof of, and Guélat and Marcotte [10] proved in detail, linear convergence of the Frank-Wolfe algorithm with away steps. However, the objective function g of $(\mathcal{D})$ does not satisfy the conditions assumed by Wolfe and by Guélat and Marcotte: it is neither boundedly concave, nor strictly (let alone strongly) concave. Instead of using their approach, we prove our result by applying Robinson's analysis [25] of upper Lipschitzian continuity to a perturbation of problem $(\mathcal{P})$.

In Section 2.3 we will incorporate a technique into the Wolfe-Atwood method which identifies and eliminates data points in the interior of the op-

timal MVEE. Finally, Section 2.4 gives some computational results of the Wolfe-Atwood algorithm for the MVEE problem with and without the elimination technique. It is perhaps surprising that the away step greatly improves the efficiency and accuracy compared with the original Khachiyan algorithm. However, without elimination and for $m \gg n$, it can be much slower than the DRN method (a specialized interior-point algorithm) with an active-set strategy in Sun and Freund [31]. Finally, for large m and n, the Wolfe-Atwood method with elimination may be the only possibility: we have solved some problems with $m = 100,000$ and $n = 500$ in less than 5 minutes, which the DRN method, even with an active-set strategy, cannot handle. This may be particularly important for applications in classification and clustering, where a minimum-volume ellipsoid algorithm to evaluate potential clusters needs to be called repeatedly as a subroutine in an overall branch-and-bound strategy. In such a case, a fast first-order method with limited accuracy may be the method of choice.

From both a theoretical and computational viewpoint, the simple first-order method of Wolfe and Atwood for the MVEE problem appears to be an attractive algorithm. Incorporation of the elimination technique further increases its value and lets us achieve high accuracy in a short time even for very large instances.

2.1 Algorithms and Analysis

Note that the objective function g of $(\mathcal{D})$ is a concave function with gradient

$$\xi(u) := \nabla g(u) = (x_i^T (XUX^T)^{-1} x_i)_{i=1}^m, \tag{2.1.1}$$

and that, with

$$u_+ := (1 - \tau)u + \tau e_j, \tag{2.1.2}$$

rank-one update formulae give

$$(XU_+X^T)^{-1} = \frac{1}{1-\tau}\left[(XUX^T)^{-1} - \frac{\tau(XUX^T)^{-1}x_j x_j^T (XUX^T)^{-1}}{1-\tau+\tau\xi_j(u)}\right] \tag{2.1.3}$$

and

$$\det XU_+X^T = (1-\tau)^{n-1}[1-\tau+\tau\xi_j(u)]\det XUX^T. \tag{2.1.4}$$

It is therefore computationally inexpensive to update ∇g after an update such as (2.1.2) and to perform a line search on g to determine the optimal τ. Indeed, the optimal step size is (e.g., see (2.19) in Khachiyan [15])

$$\tau_* = \frac{\xi_j(u)/n - 1}{\xi_j(u) - 1}. \tag{2.1.5}$$

For these reasons, applying the Frank-Wolfe algorithm [8] to $(\mathcal{D})$ is an attractive procedure, and this was proposed by Fedorov [7] and Wynn [35], the latter without the line search, in the context of optimal design. Without confusion, we can thus call this the FW algorithm.

We say H is feasible in $(\mathcal{P})$ if it is positive definite and satisfies the constraints of $(\mathcal{P})$; similarly, u is feasible in $(\mathcal{D})$ if it satisfies the constraints of $(\mathcal{D})$ and moreover has a finite objective function value, i.e., $XUX^T \succ 0$. Suppose H and u are feasible in $(\mathcal{P})$ and $(\mathcal{D})$ respectively. Then it follows that

$$n \geq \sum_i u_i x_i^T H x_i = \sum_i H \bullet x_i u_i x_i^T = H \bullet (XUX^T) = \text{Trace}(H^{1/2}XUX^TH^{1/2}).$$

Hence we have

$$\begin{aligned}
-\ln\det H - \ln\det XUX^T &= -\ln\det HXUX^T \\
&= -\ln\det H^{1/2}XUX^TH^{1/2} \\
&= -n\ln(\Pi_{j=1}^n \lambda_j)^{1/n} \\
&\geq -n\ln\left(\frac{\sum_{j=1}^n \lambda_j}{n}\right) \\
&= -n\ln\left(\frac{\text{Trace}(H^{1/2}XUX^TH^{1/2})}{n}\right) \\
&\geq 0,
\end{aligned}$$

where the λ_j's are the positive eigenvalues of $H^{1/2}XUX^TH^{1/2}$. This proves weak duality, and gives the following sufficient conditions for optimality in both ($\mathcal{P}$) and ($\mathcal{D}$):

(a) $u_i > 0$ only if $x_i^T H x_i = n$; and

(b) $H = (XUX^T)^{-1}$.

This follows since we must have all eigenvalues equal, and hence $H^{1/2}XUX^TH^{1/2}$ a multiple of the identity, to have the geometric and arithmetic means coincide, and we must have the multiple equal to unity to have the trace equal to n. In fact, it is easy as in Chapter 1 to show using the Karush-Kuhn-Tucker conditions for ($\mathcal{P}$) that these conditions are also necessary. Moreover, u provides a vector of Lagrange multipliers for ($\mathcal{P}$).

We also have (e.g., Khachiyan [15] or also from (1.3.8)) that, for any feasible u,

$$u^T\xi(u) = n,$$

so that, given (b), (a) above holds if $x_i^T H x_i \leq n$ for all i. Hence, for a feasible u, we need only check that $H = (XUX^T)^{-1}$ is feasible in ($\mathcal{P}$) to check the optimality of u.

The FW algorithm thus starts with some feasible u with XUX^T positive definite, and then at each iteration finds the index j with maximal $\xi_j(u) = x_j^T(XUX^T)^{-1}x_j$, stops if this maximum value is at most $(1+\epsilon)n$, and otherwise replaces u with u_+ in (2.1.2), where τ is chosen to maximize $g(u_+)$.

We motivated the algorithm above using the optimality conditions, but note

that e_j (the jth unit vector in $\mathbb{R}^m$) solves the problem

$$\max_{\bar{u}}\{g(u) + \nabla g(u)^T(\bar{u} - u) : e^T\bar{u} = 1, \bar{u} \geq 0\},$$

so at each iteration we maximize a linear approximation to g and do a line search on the line segment joining our current iterate to the optimal solution of the linearized problem: that is, we are performing the Frank-Wolfe algorithm on $(\mathcal{D})$.

When the algorithm stops, we have $(1+\epsilon)^{-1}H$ feasible in $(\mathcal{P})$, so that this and u are both optimal in their respective problems up to an additive constant of $n\ln(1+\epsilon) \leq n\epsilon$. Moreover, $\text{conv}\{\pm x_1, \ldots, \pm x_m\}$ is contained in $\{x \in \mathbb{R}^n : x^THx \leq (1+\epsilon)n\}$, but also contains $\{x \in \mathbb{R}^n : x^THx \leq 1\}$, since the maximum of $|v^Tx|$ over the latter is $\sqrt{v^TH^{-1}v} = \sqrt{v^TXUX^Tv} = \sqrt{\sum_i u_i(v^Tx_i)^2} \leq \max_i |v^Tx_i|$. Thus we have a $\sqrt{(1+\epsilon)n}$ rounding of this convex hull. Finally, for $0 < \eta \leq 1$, we have $n\ln(1+\epsilon) \leq 2\ln(1+\eta)$ for $\epsilon = \eta/n$, so for this value of ϵ we get an ellipsoid that has minimum volume up to the the factor $(1+\eta)$.

(Khachiyan [15] shows that to find a $(1+\epsilon)n$ rounding of the convex hull of m points $y_1, \ldots, y_m$ in $\mathbb{R}^n$, or to find a nearly minimum-volume not-necessarily-central ellipsoid containing these points, it suffices to find a good rounding or a nearly minimum-volume central ellipsoid for the set of the previous paragraph, where $x_i = (y_i; 1) \in R^{n+1}$ for each i. So at the expense of increasing the dimension by one, we can confine our attention to the central case.)

We call a feasible u *ϵ-primal feasible* if $x_i^T(XUX^T)^{-1}x_i \leq (1+\epsilon)n$ for all i, and say that it *satisfies the (strong) ϵ-approximate optimality conditions* if moreover $x_i^T(XUX^T)^{-1}x_i \geq (1-\epsilon)n$ whenever $u_i > 0$. The algorithms of Khachiyan [15] and Kumar and Yıldırım [18] seek an ϵ-primal feasible u, while that of Todd and Yıldırım [32] seeks one satisfying the ϵ-approximate optimality conditions. In

fact, apart from the details of their initialization and termination, the first two methods coincide with that of Fedorov (and Wynn, although he didn't use an optimal line search) for the optimal design problem, and hence a specialization of that of Frank and Wolfe. We therefore denote them the FW-K method and the FW-KY method.

Let us now describe the method analyzed by Todd and Yıldırım.

Algorithm 2.1.1

Input: $X \in \mathbb{R}^{n\times m}$, $\epsilon > 0$.

Step 0. *Use the algorithm of Kumar and Yıldırım [18] to obtain an initial feasible u. Compute $\xi(u)$.*

Step 1. *Find* $j := \arg\max\{\xi_l(u) - n\}$, $i = \arg\min\{\xi_l(u) - n : u_l > 0\}$.

If $\xi_j(u) - n \le \epsilon n$ *and* $\xi_i(u) - n \ge -\epsilon n$,

STOP: *u satisfies the ϵ-approximate optimality conditions.*

Else,

if $\xi_j(u) - n > n - \xi_i(u)$, ***go to*** *Step 2;*

else, go to *Step 3.*

Step 2. *Replace u as in (2.1.2), where $\tau > 0$ is chosen as in (2.1.5) to maximize g.* ***Go to*** *Step 4.*

Step 3. *Replace u by $u_+ := (1-\tau)u + \tau e_i$, where now τ is chosen from negative values to maximize g subject to u_+ remaining feasible.*

Step 4. *Update $\xi(u)$ and* ***go to*** *Step 1.*

In Step 3, the optimal unconstrained τ is again given by (2.1.5), with i replacing j, as long as $\xi_j(u) > 1$: otherwise, τ is made as negative as feasible. It is

easily seen that e_i solves the problem of minimizing the linearization of g on a restriction of the feasible set, where zero components of u are fixed at zero, so this is the FW algorithm with away steps as in Wolfe [33] (u moves away from e_i), with specific initialization and termination details given. This algorithm was also proposed by the statistician Atwood [4] for the optimal design problem. We therefore call it the WA-TY method.

Observe that ($\mathcal{P}$) can be reformulated as having a strictly convex continuous objective function and a compact feasible set, so that it has a unique optimal solution H_* with optimal value f_*, and ($\mathcal{D}$) also has an optimal solution, possibly not unique, with optimal value $g^* = f_*$. The analyses of Khachiyan [15], Kumar-Yıldırım [18], and Todd-Yıldırım [32] bound the number of steps until an ϵ-primal feasible solution u is obtained (or until one satisfying the ϵ-approximate optimality conditions is found), by bounding the improvement in $g(u)$ at each iteration.

Khachiyan starts with $u_0 = (1/m)e$, while Kumar-Yıldırım and Todd-Yıldırım start with a more complicated procedure to determine a u_0 with at most $2n$ positive components. Khachiyan shows that at most $4n(\ln n + \ln\ln m + 2)$ iterations are necessary from his initial solution until a 1-primal feasible solution is found, while Kumar and Yıldırım show that no more than $16n(\ln n + 1)$ are needed from their start to obtain the same quality. The same is true for the WA-TY method, since until a 1-primal feasible solution is obtained, no away steps will be performed. We therefore concentrate on the algorithms after they produce a 1-primal feasible solution (which also satisfies the 1-approximate optimality conditions) until they reach an ϵ-primal feasible solution or one that satisfies the ϵ-approximate optimality conditions. For this analysis, we need the following

results.

Lemma 2.1.1 *(Khachiyan [15], Lemma 2). If u is δ-primal feasible (and hence if it satisfies the δ-approximate optimality conditions),*

$$g^* - g(u) \leq n\delta. \tag{2.1.6}$$

For our analysis of away steps, it is convenient to characterize normal FW steps where u_j is increased from zero as *add*-iterations, and those where it is increased from a positive value as *increase*-iterations. Away steps are called *drop*-iterations if u_j is decreased to zero, and otherwise *decrease*-iterations. Note that every drop-iteration can be associated with either a previous add-iteration where that component of u was last increased from zero, or with one of the original at most $2n$ positive components of u_0.

Lemma 2.1.2 *Suppose $\delta \leq 1/2$.*

(a) If u is not δ-primal feasible, any add- or increase-iteration improves $g(u)$ by at least $2\delta^2/7$.

(b) If a feasible u does not satisfy the δ-approximate optimality conditions, any decrease-iteration improves $g(u)$ by at least $2\delta^2/7$.

Proof: Khachiyan [15] (Lemma 3, see also the proof of Lemma 4) proved (a), while Todd and Yıldırım [32] (Lemma 4.2) proved (b). □

Because they are limited by remaining feasible, drop-iterations may not provide a certifiably large increase in g, but at least g does not decrease.

Let $h(\delta)$ (respectively, $\bar{h}(\delta)$) denote the number of iterations of the FW-K or FW-KY method (number of add-, increase-, and decrease-iterations of the WA-TY method) from the first iterate that is δ-primal feasible (satisfies the δ-approximate optimality conditions) until the first that is $\delta/2$-primal feasible (satisfies the $\delta/2$-approximate optimality conditions). Then Lemmas 2.1.1 and 2.1.2 show that

$$h(\delta) \leq n\delta/(2(\delta/2)^2/7) = 14n/\delta, \tag{2.1.7}$$

and similarly for $\bar{h}$. So if $\mathcal{H}(\epsilon)$ (respectively, $\bar{\mathcal{H}}(\epsilon)$) denotes the number of iterations (number of add-, increase-, and decrease-iterations) from the first iterate that is 1-primal feasible (satisfies the 1-approximate optimality conditions) until the first that is ϵ-primal feasible (satisfies the ϵ-approximate optimality conditions), we find

$$\begin{aligned} \mathcal{H}(\epsilon) &\leq h(1) + h(1/2) + \cdots + h(1/2^{\lceil \ln 1/\epsilon \rceil - 1}) \\ &\leq 14n(1 + 2 + \cdots + 2^{\lceil \ln 1/\epsilon \rceil - 1}) \leq 28n/\epsilon, \end{aligned} \tag{2.1.8}$$

and again similarly for $\bar{\mathcal{H}}$. Hence we have the following

Theorem 2.1.1 *(a) The total number of iterations for the FW-K algorithm to obtain an ϵ-primal feasible solution is at most $28n/\epsilon + 4n(\ln n + \ln\ln m + 2)$, while for the FW-KY algorithm, it is at most $28n/\epsilon + 16n(\ln n + 1)$.*

(b) The total number of iterations for the WA-TY method to obtain a solution u which satisfies the ϵ-approximate optimality conditions is at most $56n/\epsilon + 32n(\ln n + 2)$.

(c) The total number of iterations for the FW-K algorithm to obtain an η-optimal solution (i.e., a solution u with $g^* - g(u) \leq \eta$) is at most $3.5n^2/\eta + 4n(\ln n + \ln\ln m + 6)$, while for the FW-KY algorithm, it is at most $3.5n^2/\eta + 16n(\ln n + 2)$ and for the WA-TY method it is at most $7n^2/\eta + 32n(\ln n + 3)$.

Proof: The argument for (a) is stated above the statement of the theorem, and for (b) we need only note that the number of drop-iterations is bounded by the number of add-iterations plus $2n$.

For part (c) we note first that if we have an η/n-primal feasible solution or one that satisfies the η/n-approximate optimality conditions, then we automatically have an η-optimal solution by Lemma 2.1.1. Thus (c) almost follows from (a) and (b). To obtain the improved coefficient for n^2/η, and to simplify the proof, we use the proof technique of Wolfe [33]. Let γ denote $g^* - g(u)$ and γ_+ denote $g^* - g(u_+)$. We obtain a $1/2$-primal feasible solution or one satisfying the $1/2$-approximate optimality conditions in $14n$ or $28n$ more steps than to find a 1-primal feasible solution or one satisfying the 1-approximate optimality conditions. From then on, $\gamma \leq n/2$ and u is not δ-primal feasible or does not satisfy the δ-approximate optimality conditions for all $\delta < \gamma/n \leq 1/2$. Then Lemmas 2.1.1 and 2.1.2 show that, at every add-, increase-, or decrease-iteration, $\gamma_+ \leq \gamma - 2\gamma^2/(7n^2)$, so if we set $\bar{\gamma}$ to be $\gamma/(3.5n^2)$ and similarly for $\bar{\gamma}_+$, we find

$$\frac{1}{\bar{\gamma}_+} \geq \frac{1}{\bar{\gamma}(1-\bar{\gamma})} \geq \frac{1+\bar{\gamma}}{\bar{\gamma}} = \frac{1}{\bar{\gamma}} + 1,$$

and so, from its initial positive value, $1/\bar{\gamma}$ will increase to at least k in k iterations; thus γ will be at most η in at most $3.5n^2/\eta$ iterations. For the WA-TY method, the bound must again be doubled for the drop-iterations. □

Observe that the more complicated analysis of Khachiyan leads to bounds on the number of iterations to be able to guarantee a certain quality solution, while the simpler argument for part (c) gives bounds on the number of iterations required to obtain a certain quality solution, but we may not know that this quality has been reached.

2.2 Local Convergence of the WA-TY Algorithm

We now wish to show that the WA-TY algorithm modification, i.e., the inclusion of decrease- and drop-iterations, leads to an asymptotic bound that grows with $\ln(1/\epsilon)$ rather than $1/\epsilon$, that is linear convergence. Unfortunately, this bound depends on the data of the problem as well as the dimensions, and so does not provide a global complexity bound better than that above.

We use the following perturbation of $(\mathcal{P})$:

$$\begin{array}{lll} & \min_{H\succ 0} & -\ln\det H \\ (\mathcal{P}(\kappa)) & & x_i^T H x_i \;\leq\; n+\kappa_i, \quad i=1,\ldots,m. \end{array}$$

Given u satisfying the δ-approximate optimality conditions, let $H(u) := (XUX^T)^{-1}$ and define $\kappa := \kappa(u,\delta) \in \mathbb{R}^m$ by

$$\kappa_i := \begin{cases} \delta n & \text{if } u_i = 0 \\ x_i^T H(u) x_i - n & \text{else.} \end{cases}$$

Observe that each component of κ has absolute value at most δn, and that this property fails if we merely assume that u is δ-primal feasible. Moreover,

$$u^T\kappa = \sum_{i:u_i>0} u_i\kappa_i = u^T\xi(u) - ne^T u = n - n = 0. \tag{2.2.9}$$

Lemma 2.2.1 *Suppose u satisfies the δ-approximate optimality conditions. Then $H(u)$ is optimal in $(\mathcal{P}(\kappa(u,\delta)))$.*

Proof: We note that $H(u)$ is feasible and that u provides the required vector of Lagrange multipliers, which suffice because the problem is convex. □

Let $\phi(\kappa)$ denote the value function, the optimal value of $(\mathcal{P}(\kappa))$. Then ϕ is convex, and if u' is any vector of Lagrange multipliers for the optimal solution

of $(\mathcal{P}(\kappa))$, then u' is a subgradient of ϕ at κ. In particular, if u_* is any vector of Lagrange multipliers for the optimal solution of $(\mathcal{P})$, then u_* is a subgradient of ϕ at 0, and we find for any u satisfying the δ-approximate optimality conditions and $\kappa := \kappa(u, \delta)$,

$$\begin{aligned} g(u) = f(H(u)) = \phi(\kappa) &\geq \phi(0) + u_*^T \kappa \\ &= g^* + (u_* - u)^T \kappa \\ &\geq g^* - \|u - u_*\|\|\kappa\|. \end{aligned} \tag{2.2.10}$$

Here the last equality follows from (2.2.9). We have already noted that $\|\kappa\| \leq n\sqrt{m}\delta$. To obtain an improvement on Lemma 2.1.1, we need to bound $\|u - u_*\|$. Robinson's second-order sufficient condition [25] requires that the Hessian of the Lagrangian be positive definite in certain directions; for linear constraints, these are just the feasible (nonzero) directions. Since f is strongly convex near any $H \succ 0$ and the constraints are linear, this condition holds at (H, u') for any $(\mathcal{P}(\kappa))$, where H is the optimal solution and u' any vector of Lagrange multipliers. Moreover, since the constraints are linear and the Mangasarian-Fromovitz constraint qualification holds (when $\|\kappa\| < 1$), the constraints are regular in the sense of Robinson at any feasible H. In addition, the constraints on H (besides the open convex set constraint that $H \succ 0$) are polyhedral, so that Robinson's Corollary 4.3 applies, which shows that, for some Lipschitz constant L, there is some u_* which is a vector of Lagrange multipliers for $(\mathcal{P})$ such that

$$\|u - u_*\| \leq L\|\kappa\| \leq Ln\sqrt{m}\delta$$

whenever $\|\kappa\|$ is sufficiently small. From this and (2.2.10) we conclude

Proposition 2.2.1 *There is some constant $M > 0$ (depending on the data of problem $(\mathcal{P})$) such that, whenever u satisfies the δ-approximate optimality conditions for some*

sufficiently small δ, we have

$$g^* - g(u) \leq M\delta^2. \tag{2.2.11}$$

Applying Proposition 2.2.1 instead of Lemma 2.1.1 in (2.1.7), we obtain

$$\bar{h}(\delta) \leq M\delta^2/(2(\delta/2)^2/7) = 14M \text{ for sufficiently small } \delta, \tag{2.2.12}$$

and this yields, using the argument above (2.1.8), the existence of a constant $Q > 0$ with

$$\bar{\mathcal{H}}(\epsilon) \leq Q + 28M \ln(1/\epsilon) \text{ for sufficiently small } \epsilon.$$

We therefore have

Theorem 2.2.1 *There are data-dependent constants $\bar{Q}$ and $\hat{Q}$ such that:*

(a) The WA-TY algorithm for problem $(\mathcal{P})$ requires at most $\bar{Q} + 56M \ln(1/\epsilon)$ iterations to obtain a solution that satisfies the ϵ-approximate optimality conditions; and

(b) The WA-TY algorithm for problem $(\mathcal{P})$ gives a sequence of optimality gaps $g^* - g(u)$ that is nonincreasing and, asymptotically, at every add-, increase-, or decrease-iteration, decreases by the factor $1 - (3.5M)^{-1}$, so that at most $\hat{Q} + 7M \ln(1/\eta)$ iterations are required to obtain an η-optimal solution.

Here M is as in Proposition 2.2.1.

Proof: Part (a) follows directly from the analysis above, again allowing for the drop-iterations. For part (b), note that asymptotically, for every add-, increase- or decrease-iteration, Lemma 2.1.2 and Proposition 2.2.1 imply that

$$g^* - g(u_+) \leq (1 - \frac{2}{7M})(g^* - g(u)),$$

which gives the result. □

To conclude this section, we observe that Proposition 2.2.1 not only is used to help prove the convergence result above, but also implies that asymptotically, solutions u that satisfy the ϵ-approximate optimality conditions are likely to be much closer to optimality than those that are merely ϵ-primal feasible, even if no improved bound can be given because M is unknown.

The local linear convergence property discussed in this section can be applied to other problems. In addition to our results in Chapters 3 and 4, Yıldırım proves a similar result for the Minimum Enclosing Ball problem in [37]. Also in [2], we prove linear convergence of the Frank-Wolfe algorithm with Wolfe's away steps [33] for the problem

$$(\mathcal{S}) \qquad \begin{array}{rcl} \max_u \quad g(u) & & \\ e^T u & = & 1, \\ u & \geq & 0, \end{array} \tag{2.2.13}$$

where g is a twice continuously differentiable concave function on the simplex.

2.3 Identification and Elimination of Interior Points

The following theorem provides a criterion to identify and eliminate points which cannot be support points of the MVEE.

Theorem 2.3.1 (Harman-Pronzato [11]) *For any ϵ-approximate optimal solution u, no point x_i such that*

$$\xi_i(u) < n[1 + \frac{\epsilon}{2} - \frac{\sqrt{\epsilon(4 + \epsilon - 4/n)}}{2}]$$

can be a support point of the MVEE, i.e., any such point x_i is an interior point of the MVEE.

Note that since we update ξ at every iteration, we can identify interior points of the MVEE in $O(m)$ time at every iteration and incorporate an elimination step into any of the algorithms mentioned above. Since it is the most promising one, we have done so for the WA-TY algorithm so that every 20 iterations points which do not satisfy this criterion are eliminated. (The extra time spent for the identification of interior points and data handling related to their elimination can be comparable to the total time spent for each iteration; hence the elimination procedure is not applied at every iteration, but once in every 20. This is a number based on our practical experience and can be tuned for each data set.) The modified version is called the WA-TY-e20 algorithm and outperforms the original version as discussed in the next section.

2.4 Computational Study

In this section we present some computational results for the Wolfe-Atwood-Todd-Yıldırım (WA-TY) modified FW algorithm, using different initialization strategies. Specifically, we test the original Khachiyan initialization strategy, where the initial feasible u is set to be the center of the simplex in $\mathbb{R}^m$, that is, $u_i = 1/m$ for all $i = 1, \ldots, m$. We also test the Kumar-Yıldırım initialization strategy introduced in [18]. The speed-up (and also possible slow-down for some extreme data sets) by the addition of the elimination technique discussed in Section 2.3 is also investigated.

We compare the above Frank-Wolfe-type first-order algorithms with a second-order interior-point algorithm, the DRN algorithm proposed in Sun and Freund [31]. For better illustration, we use the same test data sets as in Sun and

Freund [31].

In Table 2.1, we compare the computation time of the DRN algorithm and the WA-TY algorithm with the two initialization strategies on small- to medium-sized data sets. The results given are the geometric means of the solution times for 10 random problems, using the data sets from Table 2 of Sun and Freund [31]. We set $\epsilon = 10^{-7}$ for the WA-TY algorithm, and $\epsilon_1 = \epsilon_2 = 10^{-7}$ for the DRN algorithm (see Sun and Freund [31]). It is clear from the results that, while the computation time for algorithm DRN increases dramatically with the increase in the number of data points m, the running time for the WA-TY algorithm increases more or less linearly. Therefore while DRN is slightly faster for small-sized problems, the WA-TY algorithm shows a decisive advantage in large-scale problems compared to the DRN algorithm not combined with active-set strategies. However, when an active-set strategy is added to the DRN method, it becomes comparable or slightly faster than the WA-TY algorithm. Another observation is that the Kumar-Yıldırım initialization strategy demonstrates a considerable advantage over the original Khachiyan initialization strategy, especially for problems with large m.

We also tested the original FW-K and FW-KY algorithms. We stopped the algorithms after 100,000 iterations, which took from 300 to 450 seconds, at which point the optimality gap ϵ was only around 10^{-4}. It is striking that the away steps enable the FW algorithm to achieve a high degree of accuracy.

We note that [31] did not take advantage of the rank-one updating formulae for the FW-K method (called there the conditional gradient method) in the complexity analysis in the end of Section 4. The pessimistic view regarding its computation time in practice (see the end of Section 7 in [31]) is also partly due to the

same error in the implementation. Our computational experience confirms that a correctly implemented FW-KY method is able to reach low accuracy (10^{-3}) in a reasonable time for small instances, but not high accuracy (10^{-7}).

Table 2.1: Geometric mean of solution times of algorithms DRN (with and without an active-set strategy) and the WA-TY algorithm with the Kumar-Yıldırım initialization (KY) versus the Khachiyan initialization (Kha).

		Geometric Mean of Time (Seconds)			
n	m	WA-TY (KY)	WA-TY (Kha)	DRN	DRN/Act. Set
10	50	0.101	0.103	0.025	0.026
10	100	0.197	0.214	0.103	0.052
10	200	0.204	0.254	0.613	0.082
10	400	0.355	0.525	4.727	0.10
10	600	0.557	0.897	15.435	0.14
10	800	0.603	1.045	38.112	0.17
20	200	0.321	0.384	0.576	0.22
20	300	0.498	0.634	1.876	0.37
20	400	0.757	0.936	4.523	0.44
20	600	0.879	1.172	14.155	0.57
20	800	1.307	1.779	34.37	0.65
20	1000	1.289	1.982	71.292	0.66
20	1200	1.424	2.433	141.178	0.97
30	450	0.906	1.043	6.041	0.96
30	900	1.764	2.395	49.573	2.04
30	1350	2.529	3.794	187.907	2.48
30	1800	3.268	5.327	453.821	2.60

Table 2.2 demonstrates the performance of the WA-TY algorithm on larger data sets, compared with the DRN algorithm combined with an active-set strategy as in Sun and Freund [31]. Again, the results given are the geometric means of solution times and numbers of iterations for 10 random problems, now from the data sets in Table 3 of Sun and Freund [31]. For the Kumar-Yıldırım initialization strategy, it seems that the computation time grows linearly in n and m. The results also indicate that the Kumar-Yıldırım initialization is not only advantageous in theory, but also in practice, especially for large-scale problems. The superiority in the active-set strategies suggests the potential of speeding up the computations by combining the WA-TY algorithm with some active-set heuristics.

Table 2.2: Geometric mean of solution times and numbers of iterations of the WA-TY algorithm with the Kumar-Yıldırım initialization versus the Khachiyan initialization and the DRN algorithm with an active-set strategy.

Dimensions		WA-TY (KY Init.)		WA-TY (Kha. Init.)		DRN/Act. Set
n	m	Time (sec.)	# Iter.	Time (sec.)	# Iter.	Time (sec.)
20	1,000	1.24	1885.9	2.16	2974.8	0.77
10	10,000	6.06	2108.5	45.59	11943.6	0.55
20	10,000	12.84	4055.5	56.10	13828.6	2.13
20	20,000	20.07	3714.9	177.66	23755.9	2.71
20	30,000	42.87	5403.8	394.78	35328.5	3.35
30	10,000	19.60	5479.0	66.89	15137.8	7.29
30	20,000	38.32	5839.5	222.60	25941.5	8.73
30	30,000	57.98	6085.8	458.17	36032.4	9.47

We have also tested the effectiveness of the elimination technique introduced

in Section 2.3 on large data sets. Table 2.3 presents the results for the WA-TY algorithm versus the WA-TY-e20 algorithm (with elimination). KY initialization is used for all tests. The instances are randomly generated as in Sun and Freund [31]. The results given are the geometric means of solution times for 10 random problems required to obtain a 10^{-7}-approximate optimal solution. Since eliminated points do not affect steps taken by the algorithm, the number of iterations are equal for both algorithms. As the results demonstrate, the WA-TY-e20 algorithm is faster than the WA-TY algorithm and the positive effect of the elimination technique increases as the number of data points are increased.

Table 2.3: Geometric mean of solution times of the WA-TY algorithm versus the WA-TY-e20 algorithm.

Dimensions		WA-TY	WA-TY-e20
n	m	Time (sec.)	Time (sec.)
50	1000	1.4836	1.1684
50	3000	4.7456	2.5459
50	5000	10.504	4.7144
50	10000	23.372	7.8781
50	30000	81.652	20.232
50	50000	139.43	32.975
50	100000	306.01	64.647
50	300000	1159	200.3
50	500000	2260.2	364.6

In an attempt to assess the extent of extra overhead due to the elimination procedure, we considered data sets where all points lie on (or almost on) the unit sphere centered at the origin. An input set $\mathcal{A}$ is said to lie on a κ-approximate

unit sphere centered at the origin, denoted by $\mathcal{S}_\kappa$, if $\mathcal{A} \subset \mathcal{S}_\kappa := \{x \in \mathbb{R}^n : 1 - \kappa \leq \|x\| \leq 1 + \kappa\}$. For an input set $\mathcal{A} \subset \mathcal{S}_\kappa$ where $\kappa \geq 0$ is small, the elimination procedure will keep testing input points for removal once in every 20 iterations but will be unable to remove a substantial subset of the input set. In the extreme case where $\kappa = 0$, none of the input points can be removed since there would be no interior point. This extra overhead will necessarily result in an increase in the running time of an algorithm that uses the elimination procedure. We generated random input sets $\mathcal{A} \subset \mathcal{S}_\kappa$, where $\kappa \in \{0, 0.001, 0.01, 0.1, 0.2\}$, with sizes (n, m) varying from $(10, 500)$ to $(50, 50000)$. For each choice of experimental parameters, the computational results averaged over ten data sets. We used $\epsilon = 10^{-3}$. The computational results are reported in terms of averages over these instances in Table 2.4, which is divided into two sets of columns. The first set of columns reports the size (n, m). The second set of columns presents the results regarding the "slow-down" factor measured in terms of the ratio of the running time of the WA-TY method with the elimination procedure (i.e., the WA-TY-e20 method) to the running time of the same algorithm without the elimination. This set is further divided into five columns corresponding to different values of the parameter κ. A close examination of the table reveals that the slow-down factors usually remain at an acceptable level. Note that the elimination procedure leads to an extra overhead of at most 65% on all instances. A comparison of the slow-down and speed-up factors stemming from our experiments seems to justify the use of the elimination procedure especially since spherical (ellipsoidal) input sets would not likely be encountered in practical applications.

As we noted in the introduction, first-order methods such as the WA-TY algorithm can be applied to large-scale problems in high dimension n, which cannot be solved (usually because of memory limitations) by the DRN method,

Table 2.4: Experimental Results for Almost Spherical Input Sets

Dimensions		κ				
n	m	0	0.001	0.01	0.1	0.2
10	500	1.6405	0.8879	0.9842	1.1839	1.121
10	1000	1.2723	0.9110	1.0379	0.9335	1.0031
20	5000	0.9577	0.7383	0.7918	0.6187	0.5657
20	10000	0.8856	0.7425	0.7350	0.4645	0.4076
30	30000	0.9703	0.9427	0.9406	0.5704	0.4816
50	50000	1.0404	1.1455	1.162	0.9036	0.8055

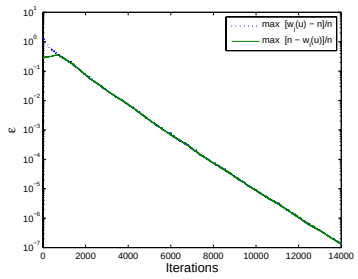

Figure 2.1: Linear Convergence of WA-TY.

even with an active-set strategy. Figure 2.1 demonstrates the typical behavior of the WA-TY algorithm on such a large data set. The curves indicate the number of iterations needed to satisfy the ϵ-approximate optimality conditions on a randomly generated data set with 10,000 points in 500-dimensional space. From Figure 2.1 we can clearly observe the linear convergence of the WA-TY algorithm, which seems to occur from the first iteration. We followed the same approach as in [31] to randomly generate 10 data sets with the same dimensions. It took the WA-TY algorithm 10 to 30 minutes (24.7 minutes on average) to solve each case to obtain a 10^{-7}-approximate optimal solution. When we

use WA-TY-e20, these problems can be solved in 4 to 13 minutes. The memory usage to solve these instances (without elimination) was generally around 166 megabytes (166M), with periodic jumps to up to 208M. As a comparison, when we used the DRN algorithm (without the active-set strategy) to solve a much smaller case with 1800 points in 30-dimensional space (one of those used to generate the last row of Table 2.1), the memory usage constantly jumped between 190M and 218M. With an active-set strategy, the DRN algorithm is able to solve some cases with 10,000 points in 100-dimensional space with similar memory usage.[1] When we attempted to run a case with dimensions 100×10,000 using the DRN algorithm without active-set strategy, Matlab crashed due to an *"Out of memory"* error, after consuming more than 2G memory. Even with an active-set strategy, Matlab crashed for the same reason on 500×10,000 cases. Our results are the first reported in the literature that solve MVEE problems of sizes as large as 500×10,000 to near-optimality, clearly demonstrating the robustness of the WA-TY algorithm in solving extremely large-scale instances of the MVEE problem. Incorporation of the elimination technique increases the attractiveness of the WA-TY algorithm even further.

[1] The number of points in an active set, and therefore the memory usage, depend on the data and implementation details of the active set algorithm.

CHAPTER 3

THE TRACE CRITERION

Let $\mathcal{X} = \{x_1, \ldots, x_m\} \subset \mathbb{R}^n$ be a set of regression vectors and X denote a matrix of order $n \times m$ whose columns consist of these vectors. Finding a design which minimizes the mean dispersion of the parameters in (1.1.4) amounts to solving

$$
\begin{array}{rrcl}
 & \max_u \ \hat{g}(u) & := & -\mathrm{Trace}(XUX^T)^{-1} \\
(\hat{\mathcal{D}}) & e^T u & = & 1, \\
 & u & \geq & 0,
\end{array}
$$

where matrix U is a diagonal matrix with the components of u on its diagonal and e is a vector of ones in $\mathbb{R}^m$ as in the previous chapters. Problem $(\hat{\mathcal{D}})$ is referred to as the A-optimal design problem in statistics. In [7], Fedorov proved that a Frank-Wolfe type algorithm converges to an optimal design and discussed the conditions under which D-optimal and A-optimal designs coincide. In this chapter, we will introduce a pair of problems dual to each other and closely related to $(\hat{\mathcal{D}})$. Using the interplay between these problems, we will develop various Frank-Wolfe type algorithms and prove that an ϵ-approximate solution (defined as in Chapter 1) can be obtained in $O(n \ln n + \epsilon^{-1})$ or $O(\ln m + \epsilon^{-1})$ iterations. Each step of the algorithm can be performed in $O(nm)$ arithmetic operations. In Section 3.3, we will prove that some of these algorithms possess a local linear convergence property. These algorithms are also preferable in practice as illustrated by the computational results in Section 3.4.

3.1 Duality and Optimality Conditions

Consider the following two problems:

$$\begin{aligned} \min \quad & f(H) := -2 \ln \text{Trace} H^{1/2} \\ (\mathcal{P}) \quad & x_i^T H x_i \leq 1, \; i = 1, \ldots, m, \end{aligned}$$

and

$$\begin{aligned} \max_u \quad & g(u) := -\ln \text{Trace}(XUX^T)^{-1} \\ (\mathcal{D}) \quad & e^T u = 1, \\ & u \geq 0. \end{aligned}$$

$(\mathcal{P})$ is a special case of $\mathcal{P}_q$ in Chapter 1 in which $q = 1/2$. From a geometric point of view, it is the problem of finding an ellipsoid which encloses all data points in $\mathcal{X}$ and has the largest sum of inverses of its semi-axes. Also $(\mathcal{D})$ is a special case of $\mathcal{D}_p$ in Chapter 1 where $p = -1$. This problem is equivalent to the statistical problem $(\hat{\mathcal{D}})$ introduced above. We will use both $(\mathcal{D})$ and $(\hat{\mathcal{D}})$ in order to develop and analyze first-order algorithms for solving all of the three problems mentioned above simultaneously. We will first establish weak duality:

Lemma 3.1.1 [Weak Duality] *We have $f(H) \geq g(u)$ for any H and u feasible in $(\mathcal{P})$ and $(\mathcal{D})$, respectively.*

Proof: Follows from Lemma 1.3.1 since $p = -1$ and $q = 1/2$ are conjugate numbers in $(-\infty, 1]$. Note that we have omitted an additive constant in the objective functions of $(\mathcal{P})$ and $(\mathcal{D})$ in this chapter unlike Chapter 1. □

We next show that having two feasible solutions H and u such that $f(H) = g(u)$ is not just sufficient but also necessary for optimality.

Theorem 3.1.1 [Strong Duality] *There exist optimal solutions H^* and u^* for problems $(\mathcal{P})$ and $(\mathcal{D})$, respectively. Furthermore, the following conditions, together with primal and dual feasibility, are necessary and sufficient for optimality in both $(\mathcal{P})$ and $(\mathcal{D})$:*

a. $H^* = \frac{(XU^*X^T)^{-2}}{\text{Trace}(XU^*X^T)^{-1}}$,

b. $x_i^T H^* x_i = 1$ *if* $u_i^* > 0$.

Proof: As in the previous lemma, the proof follows from Theorem 1.3.1 for $p = -1$ and $q = 1/2$. □

After some simplification, the necessary and sufficient conditions for u to be optimal in $(\mathcal{D})$ can be written as

(i) $\alpha_i(u^*) \leq u^{*T}\alpha(u^*)$ for all i, and

(ii) $\alpha_i(u^*) = u^{*T}\alpha(u^*)$ if $u_i^* > 0$,

where $\alpha(u) := \nabla\hat{g}(u) = (x_i^T(XUX^T)^{-2}x_i)_{i=1}^m$. We say that a feasible solution u for $(\mathcal{D})$ is *ϵ-primal feasible* if $\alpha_i(u) \leq u^T\alpha(u)(1+\epsilon)$ for all i, and say that it satisfies the *ϵ-approximate optimality conditions* or it is an *ϵ-approximate optimal solution* if moreover $\alpha_i(u) \geq u^T\alpha(u)(1-\epsilon)$ for all i such that $u_i > 0$. (Note that these definitions can be deduced from those in Chapter 1 for $p = -1$ and $q = 1/2$.)

Lemma 3.1.2 *Let u be an ϵ-primal feasible solution. Then we have*

i. $0 \leq g^* - g(u) \leq \ln(1+\epsilon)$

ii. $1 \leq \frac{\hat{g}(u)}{\hat{g}^*} \leq 1+\epsilon$,

where g^* and $\hat{g}^*$ are the optimal objective function values of $(\mathcal{D})$ and $(\hat{\mathcal{D}})$, respectively.

Proof: Since u is an ϵ-primal feasible solution, $\frac{(XUX^T)^{-2}}{(1+\epsilon)\text{Trace}(XUX^T)^{-1}}$ is feasible with respect to $(\mathcal{P})$. Let H^* and u^* be optimal solutions of $(\mathcal{P})$ and $(\mathcal{D})$, respectively. Then we have

$$\begin{aligned} -2\ln\text{Trace}\left(\frac{(XUX^T)^{-2}}{(1+\epsilon)\text{Trace}(XUX^T)^{-1}}\right)^{1/2} + 2\ln\text{Trace}H^{*1/2} &\geq 0, \text{or} \\ \ln(1+\epsilon) - \ln\text{Trace}(XUX^T)^{-1} - g(u^*) &\geq 0, \text{from which} \\ 0 \leq g^* - g(u) &\leq \ln(1+\epsilon), \end{aligned} \tag{3.1.1}$$

which proves (i). Property (ii) follows from $g = -\ln(-\hat{g})$. □

3.2 Algorithms and Analysis

In the rest of this chapter, we will develop various iterative algorithms for solving $(\mathcal{D})$ and $(\hat{\mathcal{D}})$. We will assume that the following assumption holds, i.e., $\xi(u)$ is bounded from above, for every feasible solution u produced by these algorithms:

Assumption 3.2.1 *The dual feasible variable u satisfies $\xi(u) \leq \Xi$ for some $\Xi > 1$.*

The objective function $\hat{g}$ of $(\hat{\mathcal{D}})$ is a concave function with gradient $\alpha(u)$ and that, with

$$u_+ := (1-\tau)u + \tau e_j, \tag{3.2.2}$$

rank-one update formulae give

$$\begin{aligned} \hat{g}(u_+) &= -\text{Trace}(XU_+X^T)^{-1} \\ &= -\text{Trace}\left((1+\lambda)\left((XUX^T)^{-1} - \frac{\lambda(XUX^T)^{-1}x_jx_j^T(XUX^T)^{-1}}{1+\lambda\xi_j(u)}\right)\right) \end{aligned}$$

$$
\begin{aligned}
&= -(1+\lambda)\left(\text{Trace}(XUX^T)^{-1} - \frac{\lambda \text{Trace}\left((XUX^T)^{-1}x_j x_j^T (XUX^T)^{-1}\right)}{1+\lambda\xi_j(u)}\right) \\
&= (1+\lambda)\hat{g}(u) + \frac{\lambda(1+\lambda)}{1+\lambda\xi_j(u)}\alpha_j(u), \qquad (3.2.3)
\end{aligned}
$$

where $\lambda = \frac{\tau}{1-\tau}$ and $\xi_j(u) := x_j^T(XUX^T)^{-1}x_j$ as in Chapter 2. The partial derivative of the objective function is equal to

$$
\frac{\partial \hat{g}(u_+)}{\partial \lambda} = \hat{g}(u) + \frac{\lambda^2\xi_j(u) + 2\lambda + 1}{(1+\lambda\xi_j(u))^2}\alpha_j(u). \qquad (3.2.4)
$$

Let $\hat{g}$, ξ_j, and α_j be shorthand for $\hat{g}(u)$, $\xi_j(u)$, and $\alpha_j(u)$, respectively. The numerator of the partial derivative is equal to the left-hand side of the following equation (the denominator is positive):

$$
(\xi_j^2\hat{g} + \xi_j\alpha_j)\lambda^2 + \lambda(2\xi_j\hat{g} + 2\alpha_j) + \hat{g} + \alpha_j = 0. \qquad (3.2.5)
$$

We can find the best step size τ^* (or λ^*) by investigating the roots of the quadratic equation (3.2.5) and the boundary condition ($\lambda^* \geq -u_j$) arising from the nonnegativity of the dual feasible solutions as follows:

- if we have either $\xi_j\hat{g} + \alpha_j = 0$ or $(1-\xi_j)(\alpha_j + \xi_j\hat{g}) \leq 0$ (which is equivalent to $\xi_j \leq 1$ since $\alpha_j + \xi_j\hat{g} \leq 0$ for any feasible solution),
  - and if $\hat{g} + \alpha_j \geq 0$, then $\lambda^* = \infty$ and hence $\tau^* = 1$,
  - else (i.e., $\hat{g} + \alpha_j < 0$), $\lambda^* = -u_j$;
- otherwise λ^* is equal to one of the roots of the quadratic (3.2.5), which are

$$
\begin{aligned}
\lambda^*_{1,2} &= \frac{-\xi_j\hat{g} - \alpha_j \pm \sqrt{(\xi_j\hat{g} + \alpha_j)^2 - (\xi_j^2\hat{g} + \xi_j\alpha_j)(\hat{g} + \alpha_j)}}{(\xi_j^2\hat{g} + \xi_j\alpha_j)} \\
&= \frac{-\xi_j\hat{g} - \alpha_j \pm \sqrt{\alpha_j(1-\xi_j)(\alpha_j + \xi_j\hat{g})}}{(\xi_j^2\hat{g} + \xi_j\alpha_j)},
\end{aligned}
$$

  or $-u_j$ which is feasible and gives the greatest improvement in the objective function.

Once we find the step size, we can calculate $\xi(u_+)$ and $\alpha(u_+)$ from

$$\begin{aligned}
\xi_i(u_+) &= x_i^T(XU_+X^T)^{-1}x_i \\
&= x_i^T\left((1+\lambda)\left((XUX^T)^{-1} - \frac{\lambda(XUX^T)^{-1}x_jx_j^T(XUX^T)^{-1}}{1+\lambda\xi_j(u)}\right)\right)x_i \\
&= (1+\lambda)\xi_i(u) - \frac{(1+\lambda)\lambda}{1+\lambda\xi_j(u)}\xi_{ij}(u)^2 \\
&= (1+\lambda)(\xi_i(u) - \eta\xi_{ij}(u)^2), \qquad (3.2.6)
\end{aligned}$$

and

$$\begin{aligned}
\alpha_i(u_+) &= x_i^T(XU_+X^T)^{-2}x_i \\
&= x_i^T((1+\lambda)\left((XUX^T)^{-1} - \frac{\lambda(XUX^T)^{-1}x_jx_j^T(XUX^T)^{-1}}{1+\lambda\xi_j(u)}\right)... \\
&\quad (1+\lambda)\left((XUX^T)^{-1} - \frac{\lambda(XUX^T)^{-1}x_jx_j^T(XUX^T)^{-1}}{1+\lambda\xi_j(u)}\right))x_i \\
&= (1+\lambda)^2x_i^T((XUX^T)^{-2} - \frac{2\lambda}{1+\lambda\xi_j(u)}(XUX^T)^{-2}x_jx_j^T(XUX^T)^{-1}... \\
&\quad + \frac{\lambda^2}{(1+\lambda\xi_j(u))^2}(XUX^T)^{-1}x_jx_j^T(XUX^T)^{-2}x_jx_j^T(XUX^T)^{-1})x_i \\
&= (1+\lambda)^2\alpha_i(u) - 2\frac{(1+\lambda)^2\lambda}{1+\lambda\xi_j(u)}\xi_{ij}(u)\alpha_{ij}(u) + \frac{(1+\lambda)^2\lambda^2}{(1+\lambda\xi_j(u))^2}\xi_{ij}(u)^2\alpha_j(u) \\
&= (1+\lambda^2)(\alpha_i(u) - 2\eta\xi_{ij}(u)\alpha_{ij}(u) + \eta^2\xi_{ij}(u)^2\alpha_j(u)), \qquad (3.2.7)
\end{aligned}$$

where $\eta := \frac{\lambda}{1+\lambda\xi_j(u)}$, $\xi_{ij}(u) := x_i^T(XUX^T)^{-1}x_j$, and $\alpha_{ij}(u) := x_i^T(XUX^T)^{-2}x_j$. Note that all updates can be performed cheaply (in $O(nm)$ operations) as in Chapter 2; however the hidden constants in this chapter are approximately two times larger than those in the previous chapter.

Now we will describe two Frank-Wolfe type algorithms. The first algorithm (Algorithm 3.2.1) uses positive step sizes and seeks an ϵ-primal feasible solution; whereas the second one (Algorithm 3.2.2) may also have negative step sizes and stops when an ϵ-approximate optimal solution is found. This algorithm is an extension of the first one with Wolfe's away steps. We will show that

although these algorithms have similar global complexity results, away steps are necessary in order to achieve high accuracy, a phenomenon we have also observed for the MVEE problem in Chapter 2.

Algorithm 3.2.1

Input: $X \in \mathbb{R}^{n\times m}$, $\epsilon > 0$.

Step 0. *Let* $u = (1/m)e$. *Compute* $\xi(u)$ *and* $\alpha(u)$.

Step 1. *Find* $j := \arg\max_l\{\alpha_l(u) - u^T\alpha(u)\}$.

If $\frac{\alpha_j(u)}{u^T\alpha(u)} - 1 \leq \epsilon$,

STOP: u *is an* ϵ*-primal feasible solution.*

Step 2. *Replace* u *as in (3.2.2), where* $\tau > 0$ *is chosen to maximize* g.

Step 3. *Update* $\xi(u)$ *and* $\alpha(u)$. ***Go to*** *Step 1.*

Algorithm 3.2.2

Input: $X \in \mathbb{R}^{n\times m}$, $\epsilon > 0$.

Step 0. *Let* $u = (1/m)e$. *Compute* $\xi(u)$ *and* $\alpha(u)$.

Step 1. *Find* $j := \arg\max_l\{\alpha_l(u) - u^T\alpha(u)\}$ *and* $i := \arg\min\{\alpha_l(u) - e^T\alpha(u) : u_l > 0\}$.

If $\frac{\alpha_j(u)}{u^T\alpha(u)} - 1 \leq \epsilon$ *and* $1 - \frac{\alpha_i(u)}{u^T\alpha(u)} \leq \epsilon$,

STOP: *u is an* ϵ*-approximate optimal solution.*

Else,

if $\alpha_j(u) - u^T\alpha(u) > u^T\alpha(u) - \alpha_i(u)$, ***go to*** *Step 2;*

else, go to *Step 3.*

Step 2. *Replace u as in (3.2.2), where* $\tau > 0$ *is chosen to maximize g.* ***Go to*** *Step 4.*

Step 3. *Replace u by* $u_+ := (1 - \tau)u + \tau e_i$, *where now* τ *is chosen from negative values to maximize g subject to* u_+ *remaining feasible.*

Step 4. *Update* $\xi(u)$ *and* $\alpha(u)$. ***Go to*** *Step 1.*

If we look closely at these algorithms, we can identify three different types of iterations. Let u^l be the dual feasible solution at hand at iteration number l, e_{j_l} be the vertex that we use in our update and τ_l be the step size associated with this update. We refer to iteration l as

- an *add/increase step* if $\tau_l > 0$,
- a *decrease step* if $u^l_{j_l} > 0$ and $\frac{-u^l_{j_l}}{1-u^l_{j_l}} < \tau_l < 0$, and
- a *drop step* if $u^l_{j_l} > 0$ and $\tau_l = \frac{-u^l_{j_l}}{1-u^l_{j_l}}$.

We only have add/increase steps in Algorithm 3.2.1, whereas all types of steps can be performed in Algorithm 3.2.2. Note that after a drop step we have $u^{l+1}_{j_l} =$

0. In such a step, we may not be able to improve the objective function as much as we desire. Fortunately, the number of drop steps is bounded above by the number of add steps plus a constant (the number of positive components of the initial solution), and hence studying only the first two types of steps will be enough to obtain convergence results.

Lemma 3.2.1 *$u^0 = (1/m)e = \frac{1}{m}(1, 1, \ldots, 1)$ is an $(m-1)$-primal feasible solution.*

Proof: We have

$$\sum_i u_i \frac{x_i^T (XUX^T)^{-2} x_i}{\text{Trace}(XUX^T)^{-1}} = \frac{\text{Trace}(XUX^T)^{-1}}{\text{Trace}(XUX^T)^{-1}} = 1,$$

for any dual feasible solution u. In particular, for $u^0 = \frac{1}{m}(1, 1, \ldots, 1)$ we have

$$\begin{aligned} \sum_i \frac{1}{m} \frac{x_i^T (XU^0X^T)^{-2} x_i}{\text{Trace}(XU^0X^T)^{-1}} &= 1, \text{ and hence} \\ \max x_i^T (XU^0X^T)^{-2} x_i &\leq m\text{Trace}(XU^0X^T)^{-1}, \end{aligned}$$

where $U^0 = \text{Diag}(u^0)$. Hence u^0 is an $(m-1)$-primal feasible solution. □

We now analyze the first algorithm closely:

Lemma 3.2.2 *As long as u_l satisfy Assumption 3.2.1 for all $l = 1, 2, \ldots$, Algorithm 3.2.1 finds an ϵ-primal feasible solution in at most*

$$\mathcal{L}(\epsilon) = O(\ln m + \epsilon^{-1}) \tag{3.2.8}$$

steps. The constants hidden in the 'big oh' are linearly dependent on the constant Ξ in Assumption 3.2.1.

Proof: We will first prove that

$$\mathcal{L}(1) = \min\{l|\epsilon_l \leq 1\} = O(\ln m). \tag{3.2.9}$$

Let j_l be the index of the pivot point at iteration l, τ_l be the step size, and $\lambda_l = \frac{\tau_l}{1-\tau_l}$. At each iteration l with $\epsilon_l \geq 1$, from (3.2.3), we have

$$\begin{aligned}
\hat{g}(u^{l+1}) - \hat{g}(u^l) &= \lambda_l \hat{g}(u^l) + \frac{\lambda_l(1+\lambda_l)}{1+\lambda_l \xi_{j_l}(u^l)} \alpha_{s_l} \\
&\geq \frac{1}{2\xi_{j_l}(u^l)} \hat{g}(u^l) - \frac{\frac{1}{2\xi_{j_l}(u^l)}}{1+\frac{1}{2\xi_{j_l}(u^l)}\xi_{j_l}(u^l)} 2\hat{g}(u^l) \\
&\geq \frac{1}{2\xi_{j_l}(u^l)} \hat{g}(u^l) \left(1 - \frac{2}{1+\frac{1}{2}}\right) \\
&\geq -\frac{\hat{g}(u^l)}{6\Xi}.
\end{aligned} \tag{3.2.10}$$

The first inequality follows since the improvement obtained from choosing the best step length is at least as good as the improvement obtained by using any step length; in particular, it can be bounded by plugging in $\lambda_l = \frac{1}{2\xi_{j_l}(u^l)}$.

Hence we have

$$\hat{g}(u^{l+1}) \geq (1 - \frac{1}{6\Xi})\hat{g}(u^l). \tag{3.2.11}$$

Using Lemmas 3.1.2 and 3.2.1,

$$\hat{g}(u^0) \geq m\hat{g}^*. \tag{3.2.12}$$

Combining inequalities (3.2.11) and (3.2.12), we obtain

$$\hat{g}^* \geq \hat{g}(u^l) \geq \quad (1 - \tfrac{1}{6\Xi})^l \hat{g}(u^0) \geq (1 - \tfrac{1}{6\Xi})^l m\hat{g}^* \geq e^{-\frac{l}{6\Xi}} m\hat{g}^*. \tag{3.2.13}$$

Hence we must have $\mathcal{L}(1) \leq 6\Xi \ln(m) = O(\ln m)$.

Now assume that $\epsilon_l \leq 1$ and define $h(\epsilon_l) := \min\{h|\epsilon_{l+h} \leq \epsilon_l/2\}$. As long as $\epsilon_{l+h} \geq \epsilon/2$, from (3.2.3) we also have

$$\hat{g}(u^{l+h+1}) - \hat{g}(u^{l+h}) \quad \geq \quad \hat{g}(u^{l+h}) \frac{\epsilon_l}{4\xi_{l+h}(u^l)} \left(1 - \frac{1+\epsilon_l/2}{1+\frac{\epsilon_l}{4\xi_{l+h}(u^l)}\xi_{l+h}(u^l)}\right)$$

$$\geq \quad -\frac{\epsilon_l^2}{32\Xi}\hat{g}^*. \tag{3.2.14}$$

Again, the first inequality is obtained by setting $\lambda_l = \frac{\epsilon_l}{4\xi_{l+h}(u^l)}$. On the other hand, Lemma 3.1.2 gives

$$\frac{\hat{g}(u^l)}{\hat{g}^*} \leq 1 + \epsilon_l. \tag{3.2.15}$$

Combining equations (3.2.14) and (3.2.15), we get $h(\epsilon_l) \leq \frac{32\Xi}{\epsilon_l}$. Therefore

$$\begin{aligned} \mathcal{H}(\epsilon) &= h(\epsilon_l) + h(\epsilon_l/2) + h(\epsilon_l/4) + \ldots + h(\epsilon_l/2^{\lceil \ln \epsilon_l/\epsilon \rceil - 1}) \\ &\leq 32\Xi\left(\frac{1}{\epsilon_l} + \frac{2}{\epsilon_l} + \frac{4}{\epsilon_l} + \ldots + \frac{2^{\lceil \ln \epsilon_l/\epsilon \rceil - 1}}{\epsilon_l}\right) \leq \frac{64\Xi}{\epsilon} = O(\epsilon^{-1}), \end{aligned} \tag{3.2.16}$$

iterations are required to obtain an ϵ-primal feasible solution starting with a solution $\epsilon_l \leq 1$. Combining (3.2.16) and (3.2.9) completes the proof. □

Once we take care of the drop steps, the analysis of the algorithm with away steps is no more complicated.

Lemma 3.2.3 *As long as u_l satisfy Assumption 3.2.1 for all $l = 1, 2, \ldots$, Algorithm 3.2.2 finds an ϵ-approximate optimal solution in at most*

$$\mathcal{L}(\epsilon) = O(m + \epsilon^{-1}) \tag{3.2.17}$$

steps. The constants hidden in the 'big oh' are linearly dependent on the constant Ξ in Assumption 3.2.1.

Proof: We can only have add/increase steps when $\epsilon_l \geq 1$; hence Algorithms 3.2.1 and 3.2.2 take the same steps until the first solution $u_{\hat{l}}$ with $\epsilon_{\hat{l}} \leq 1$ is encountered. So that

$$\mathcal{L}(1) = \min\{l | \epsilon_l \leq 1\} = O(\ln m) \tag{3.2.18}$$

holds for Algorithm 3.2.2 as well.

Now assume that $\epsilon_l \leq 1$ and define $h(\epsilon_l) := \min\{h|\epsilon_{l+h} \leq \epsilon_l/2\}$ as before. Let us look at the improvement in the objective function at the $(l+h)^{th}$ iteration. There are three cases:

1. If this is an add/increase step, then

$$\hat{g}(u^{l+h+1}) - \hat{g}(u^{l+h}) \quad \geq \quad -\frac{\epsilon_l^2}{32\Xi}\hat{g}^* \tag{3.2.19}$$

   from (3.2.14);

2. if it is a decrease step, we have

$$\begin{aligned} \hat{g}(u^{l+h+1}) - \hat{g}(u^{l+h}) \quad &\geq \quad \hat{g}(u^{l+h})\frac{-\epsilon_l}{4\xi_{l+h}}\left(1 - \frac{1-\epsilon_l/2}{1-\frac{\epsilon_l}{4\xi_{l+h}}\xi_{l+h}}\right) \\ &\geq \quad -\frac{\epsilon_l^2}{16\Xi}\hat{g}^*; \end{aligned} \tag{3.2.20}$$

3. otherwise (it is a drop step), we can only conclude that

$$\hat{g}(u^{l+h+1}) - \hat{g}(u^{l+h}) \quad \geq 0. \tag{3.2.21}$$

Hence we have

$$\hat{g}(u^{l+h+1}) - \hat{g}(u^{l+h}) \quad \geq \quad -\frac{\epsilon_l^2}{32\Xi}g^*, \tag{3.2.22}$$

whenever we have an add/increase or decrease step.

On the other hand, using Lemma 3.1.2 we have

$$\frac{\hat{g}(u^l)}{\hat{g}^*} \leq 1 + \epsilon_l. \tag{3.2.23}$$

Combining equations (3.2.22) and (3.2.23), we need to perform at most

$$h(\epsilon_l) \leq \frac{32\Xi}{\epsilon_l}$$

add/increase and decrease steps to obtain an $\epsilon_l/2$-approximate optimal solution starting with an ϵ_l-approximate optimal solution. Applying this argument repeatedly, we conclude that we need at most

$$\begin{aligned}\mathcal{H}(\epsilon) &= h(\epsilon_l) + h(\epsilon_l/2) + h(\epsilon_l/4) + \ldots + h(\epsilon_l/2^{\lceil \ln \epsilon_l/\epsilon \rceil - 1}) \\ &\leq 32\Xi\left(\frac{1}{\epsilon_l} + \frac{2}{\epsilon_l} + \frac{4}{\epsilon_l} + \ldots + \frac{2^{\lceil \ln \epsilon_l/\epsilon \rceil - 1}}{\epsilon_l}\right) \\ &\leq \frac{64\Xi}{\epsilon} = O(\epsilon^{-1}), \end{aligned} \tag{3.2.24}$$

add/increase and decrease iterations to obtain an ϵ-approximate optimal solution starting with an ϵ_l-approximate optimal solution where $\epsilon_l \in (0, 1]$. Since the number of drop steps is bounded above by the number of add steps plus m (the number of positive components of the initial solution u^0), (3.2.17) is immediate. □

The following lemma shows that (for the same set of data points) an approximate solution to the D-optimal design problem is also close to the optimal solution of the A-optimal design problem in some sense.

Lemma 3.2.4 *Let u^D be a δ-primal feasible solution for the D-optimal design (as defined as in Chapter 2), then u^D is an $(n + n\delta - 1)$-primal feasible solution for $(\mathcal{D})$.*

Proof: For all $1 \leq j \leq m$, we have

$$\begin{aligned} x_j^T (XU^D X^T)^{-2} x_j &= \text{Trace}((XU^D X^T)^{-1}(XU^D X^T)^{-1/2} x_j x_j^T (XU^D X^T)^{-1/2}) \\ &\leq \text{Trace}((XU^D X^T)^{-1})\text{Trace}((XU^D X^T)^{-1/2} x_j x_j^T (XU^D X^T)^{-1/2}) \\ &\leq \text{Trace}((XU^D X^T)^{-1})\text{Trace}(x_j^T (XU^D X^T)^{-1} x_j) \\ &\leq \text{Trace}((XU^D X^T)^{-1})(n + n\delta), \end{aligned}$$

where $U^D = \text{Diag}(u^D)$. This proves that u^D is an $(n+n\delta-1)$-primal feasible solution for $(\mathcal{D})$. □

Let us call the algorithm which finds a 1-approximate optimal solution for the D-optimal design problem using WA-TY method of Chapter 2 and proceeds with Steps 1, 2, and 3 of Algorithm 3.2.1 as Algorithm 3.2.1-MV; and that proceeds with Steps 1, 2, 3, and 4 of Algorithm 3.2.2 as Algorithm 3.2.2-MV. When $m \gg n$, these algorithms perform significantly better than the original ones as the following lemma suggests. In addition, we are able to obtain core-set results for free.

Lemma 3.2.5 *As long as u_l satisfy Assumption 3.2.1 for all $l = 1, 2, \ldots$,*

a. *Algorithm 3.2.1-MV finds an ϵ-primal feasible solution in at most*

$$\mathcal{L}(\epsilon) = O(n \ln n + \epsilon^{-1}) \tag{3.2.25}$$

steps;

b. *Algorithm 3.2.2-MV finds an ϵ-approximate optimal solution in at most*

$$\mathcal{L}(\epsilon) = O(\min\{m, n \ln n\} + \epsilon^{-1}) \tag{3.2.26}$$

steps;

c. *furthermore, Algorithm 3.2.1-MV identifies a set $\mathcal{A} \subset \mathcal{X}$ such that*

$$|\mathcal{A}| \leq O(n \ln n + \epsilon^{-1})$$

and an ϵ-primal feasible solution u for the A-optimal design problem defined over data set $\mathcal{A}$ is also an ϵ-primal feasible solution for the A-optimal design problem defined over data set $\mathcal{X}$; and

d. *Algorithm 3.2.2-MV identifies a set $\mathcal{A} \subset \mathcal{X}$ such that*

$$|\mathcal{A}| \leq O(n \ln n + \epsilon^{-1})$$

and an ϵ-approximate optimal solution u for the A-optimal design problem defined over data set $\mathcal{A}$ is also an ϵ-approximate optimal solution for the A-optimal design problem defined over data set $\mathcal{X}$.

Proof: Part (b) of Theorem 2.1.1 shows that we can obtain a 1-approximate optimal solution for the D-optimal design problem in $O(n \ln n)$ iterations. Let u^0 be such a solution. Lemmas 3.1.2 and 3.2.4 give

$$\hat{g}(u^0) \geq 2n\hat{g}^*. \tag{3.2.27}$$

Replacing (3.2.12) with (3.2.27) in the proof of Lemma 3.2.2, gives $\mathcal{L}(1) = O(\ln n)$ for Algorithm 3.2.1-MV. Since the rest of the proof is unchanged, Algorithm 3.2.1-MV finds an ϵ-primal feasible solution in $\mathcal{L}(\epsilon) = O(n \ln n + \ln n + \epsilon^{-1}) = O(n \ln n + \epsilon^{-1})$ iterations, which proves (a).

Similarly, (b) follows from Lemma 3.2.3 with replacing $\mathcal{L}(1) = O(\ln n)$ and noticing that the number of positive components in u^0 is bounded above by $O(\min\{m, n \ln n\})$ as proved in [18].

Let $\hat{u}$ be the output of Algorithm 3.2.1-MV. Letting $\mathcal{A} = \{x_i : \hat{u}_i > 0\}$ proves (c) since the number of positive components of $\hat{u}$ is bounded above by the number of positive components in the initial solution (which is $2n$ as discussed in [18]) plus the number of add steps (which is less than the total number of iterations proved in part (a)). Similar arguments can be used to prove part (d). □

3.3 Local Convergence Properties

In this section, we will show that Algorithms 3.2.2 and 3.2.2-MV are locally linearly convergent, i.e., the number of iterations grows with $O(\ln \epsilon^{-1})$ not $O(\epsilon^{-1})$

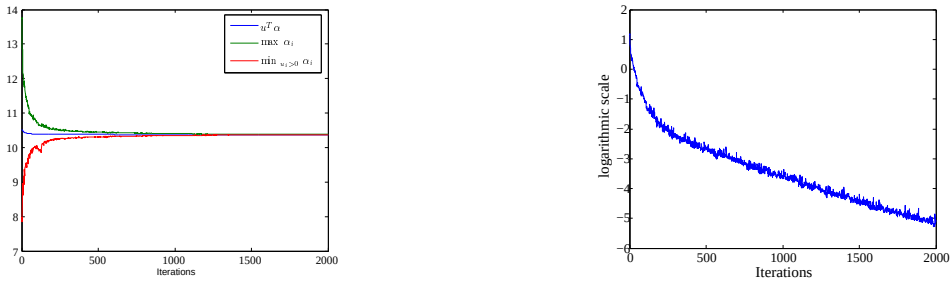

Figure 3.1: Behavior of Algorithm 3.2.2 for $(m, n) = (10000, 100)$.

asymptotically under certain assumptions. The typical behavior of the algorithms as demonstrated in Figure 3.1 illustrates this property. Unfortunately, this bound depends on the data of the problem as well as the dimensions and the constant Ξ defined as in Lemma 3.2.3, and so does not provide global complexity bounds better than those above.

Let us look at the following perturbation of the primal problem $(\mathcal{P})$:

$$\begin{array}{rlcl} & \min & f(H) & := & -2\ln \mathrm{Trace} H^{1/2} \\ (\mathcal{P}(\kappa)) & & x_i^T H x_i & \leq & 1+\kappa_i,\ i=1,\ldots,m. \end{array}$$

Given u satisfying the ϵ-approximate optimality conditions, let $H(u) := \frac{(XUX^T)^{-2}}{\mathrm{Trace}(XUX^T)^{-1}}$ and define $\kappa := \kappa(u, \epsilon)$ as

$$\kappa_i(u,\epsilon) := \begin{cases} \epsilon & \text{if } u_i = 0, \\ x_i^T H(u) x_i - 1 & \text{else.} \end{cases}$$

Note that, each component of perturbation vector κ is absolutely bounded by ϵ and $u^T\kappa = \frac{\sum_{j:u_j>0} u_j x_j^T (XUX^T)^{-2} x_j}{\text{Trace}(XUX^T)^{-1}} - 1 = 1 - 1 = 0$. $H(u)$ is optimal w.r.t. $\mathcal{P}(\kappa(u,\epsilon))$, since it is feasible and u provides the corresponding Lagrangian multipliers. Let $\phi(\kappa)$ be the value function, the optimal value of $(\mathcal{P}(\kappa))$. If u^* is a vector of multipliers corresponding to the optimal solution of $(\mathcal{P})$, then u^* is a subgradient of ϕ at 0. For any ϵ-approximate optimal solution u and $\kappa := \kappa(u,\epsilon)$, we have

$$\begin{aligned} g(u) = f(H(u)) &= \phi(\kappa) \geq \phi(0) + u^{*T}\kappa \\ &= g^* + (u^* - u)^T\kappa \geq g^* - \|u - u^*\|\|\kappa\|. \end{aligned} \tag{3.3.28}$$

Since $f(H)$ is strongly convex near any $H \succ 0$ and the constraints are linear, Robinson's second order condition holds at $(H, \hat{u})$ for any $\mathcal{P}(\kappa)$, where H is the optimal solution and $\hat{u}$ is any Lagrangian multiplier. Moreover, the linear constraints are regular at any feasible point and they are polyhedral, therefore Robinson's Corollary 4.3 applies, which shows that

$$\|u - u^*\| \leq L\|\kappa\| \leq L\sqrt{m}\epsilon,$$

where L is a data-dependent constant and whenever $\|\kappa\|$ is sufficiently small. Hence we conclude

$$g^* - g(u) \leq M\epsilon^2 \tag{3.3.29}$$

for some M depending on the data of the problem $(\mathcal{P})$. Using inequality (3.3.29), we can find a constant $\hat{c}$ such that

$$\frac{\hat{g}(u^l)}{\hat{g}^*} \leq e^{M\epsilon_l^2} \leq 1 + \hat{c}\epsilon_l^2, \tag{3.3.30}$$

for any ϵ_l-approximate solution u_l, as long as ϵ_l is small enough. Using (3.3.30) instead of (3.2.23) in the last part of the proof of Lemma 3.2.3 we obtain the following lemma:

Lemma 3.3.1 *Under the assumption of Lemma 3.2.3, there exists a data-dependent constant Q such that Algorithms 3.2.2 and 3.2.2-MV discussed above converges to an ϵ-approximate optimal solution in $O(Q + \ln(1/\epsilon))$ steps.*

3.4 Computational Study

In this section we present some computational results for Algorithms 3.2.1 and 3.2.2, using different initialization strategies: the Khachiyan initialization (KH) strategy, where the initial feasible solution u is the center of the unit simplex, i.e., $u_i = 1/m$ for all $i = 1, \ldots, m$; the Kumar-Yıldırım initialization (KY) strategy introduced in [18]; and a new strategy (MV) where the initial solution is set to be a 1-approximate optimal solution obtained by the WA-TY method of Chapter 2. All experiments were carried out on a 3.40 GHz Pentium IV processor with 1.0 GB RAM using MATLAB version R2006b.

In Table 3.1, we compare the computation time of the algorithms described above with three initializations on small- to medium-sized data sets. The data sets are generated as in [31]. The results presented are the geometric means of the solution times for 10 random problems to obtain an ϵ-primal feasible (for Algorithm 3.2.1) or an ϵ-approximate optimal solution (for Algorithm 3.2.2) where $\epsilon = 10^{-3}$. It is clear from the results that Algorithm 3.2.2 preforms significantly better than Algorithm 3.2.1 showing that away steps are necessary for develop-

Table 3.1: Geometric mean of solution times of Algorithms 3.2.1 and 3.2.2 for small-medium sized problems with different initializations

		Geometric Mean of Time (Seconds)					
		Algorithm 3.2.1			Algorithm 3.2.2		
n	m	Kha	KY	MV	Kha	KY	MV
10	100	10.5	10.3	10.1	1.2	1.3	1.9
10	200	10.8	9.9	10.6	0.6	1.4	1.1
10	400	11.9	11.2	12.5	0.4	0.8	1.0
10	600	13.3	13.0	12.7	0.6	1.1	0.8
10	800	13.9	13.4	13.4	1.0	1.5	1.2
20	200	37.9	36.4	35.3	1.2	0.8	0.6
20	300	39.6	40.0	39.2	1.4	1.1	1.0
20	400	38.3	38.5	39.7	0.7	1.7	1.6
20	600	49.2	49.2	45.7	0.9	2.0	2.9
20	800	52.6	54.5	52.3	1.2	2.5	3.4
20	1000	57.1	54.4	53.1	1.7	3.4	3.4
20	1200	58.7	56.4	56.6	1.8	5.3	5.0
30	450	108.6	100.1	93.9	2.0	2.9	2.8
30	900	130.0	119.6	127.5	1.5	4.7	4.5
30	1350	142.3	121.3	120.9	2.3	6.5	5.8
30	1800	154.2	131.3	128.9	3.5	7.6	7.7

ing efficient algorithms. For these instances, it is hard to make conclusions on the performances of the initialization strategies.

Table 3.2 presents the performance of the algorithms on larger data sets.

Table 3.2: Geometric mean of solution times of Algorithms 3.2.1 and 3.2.2 for large problems with different initializations

		Geometric Mean of Time (Seconds)					
		Algorithm 3.2.1			Algorithm 3.2.2		
n	m	Kha	KY	MV	Kha	KY	MV
5	10000	17.267	12.208	11.641	35.236	3.5327	3.5428
5	20000	26.57	20.417	20.905	55.491	7.8292	7.4747
5	30000	35.941	29.808	30.374	43.136	7.9607	9.8677
5	50000	58.433	54.698	52.828	98.456	28.159	28.715
10	10000	43.677	32.431	32.173	38.017	5.7187	5.5486
10	20000	76.886	67.377	66.554	138.93	10.604	10.154
10	30000	103.56	87.166	90.091	126.69	17.158	15.499
20	10000	141.76	113.23	117.45	48.849	18.482	19.234
20	20000	211.44	186.48	183.35	196.31	40.659	39.256
20	30000	287.15	253.81	252.65	385.37	53.223	45.749
20	50000	426.9	395.6	402.68	543.22	99.232	91.305
30	10000	295.09	247.77	243.47	59.061	27.439	31.508
30	20000	451.68	395.66	402.26	220.01	74.113	61.231
30	30000	606.04	536.8	528.98	500.77	89.2	96.194
50	50000	2308.2	2154.5	2142.8	1992.3	370.77	327.79

Again, the results are the geometric means of the solution times of 10 random problems generated as in [31] for each parameter set. The results indicate that for these instances where $m \gg n$, the MV initialization is outperforming the Khachiyan initialization as Lemma 3.2.5 suggests. Since the KY initialization is somehow close to the MV initialization, its performance it similar to the MV

Table 3.3: Geometric mean of solution times of Algorithm 3.2.2-MV with different (update) selection strategies for small instances

		Time (Seconds)		Iterations	
n	m	ALL	Orig.	ALL	Orig.
20	200	0.54	0.85	510.7	1697.9
20	300	0.67	1.16	638.5	2252
20	400	0.91	1.72	772.08	3122
20	600	1.45	2.02	904.9	3254
20	800	2.01	2.57	1028.9	3918.6
20	1000	2.67	3.41	1189.9	4836.6
20	1200	3.00	5.35	1195.3	6397
30	450	1.26	2.90	963.3	4467.3
30	900	2.82	4.71	1314.6	5723.7
30	1350	4.68	6.59	1660.4	6976.3
30	1800	6.33	7.67	1782.9	7706.8
20	1000	2.54	3.49	1168.4	4694.9

initialization. One should not be surprised by the fact that Algorithm 3.2.2 with the Khachiyan initialization is very slow on these instances, since the initial solution has many entries with positive weights and the algorithm needs to take many drop steps before converging to the optimal solution. Fortunately, other two initializations are able to find accurate solutions in short time. We have tried even larger data sets to explore the limits of the algorithms. We were able to find $10^{-}4$-approximate optimal solutions to instances where $n = 500$ and $m = 10000$ (generated as before) with Algorithm 3.2.2 using KY initialization under 30 minutes.

Table 3.4: Geometric mean of solution times of Algorithm 3.2.2-MV with different (update) selection strategies for large instances

		Time (Seconds)		Iterations	
n	m	ALL	Orig.	ALL	Orig.
10	10000	13.33	5.71	875.8	2656.5
20	10000	26.08	18.48	1634.5	6072.5
20	20000	59.32	40.61	1879.8	6852.7
20	30000	102.14	62.41	2220.3	7854.7
30	10000	42.95	27.43	2547.9	7100.8
30	20000	101.86	74.11	3085.6	10515
30	30000	140.42	89.2	2876.5	8899.2
50	50000	428.3	370.7	5106.4	15979

The number of iterations required can be significantly decreased if we make the best possible update (not just one of the two arguments used in Step 1) at each iteration. This can be done by calculating the improvement related to each index and choosing the best. We have coded a version of Algorithm 3.2.2-MV and experimented on some of the data sets above. The (mean) solution times and number of iterations are compared in Tables 3.3 and 3.4. The unmodified version of the algorithm is represented in the columns labeled with ''Orig.'' while the version with optimal decisions is labeled with ''ALL''. It is obvious that as the number of points in the data set increase calculating the possible improvement for each index becomes expensive; hence considering only two promising vertices is a wise choice. Obviously some hybid versions, which choose the best of a small set of carefully selected indices, can perform better for certain instances; so can other versions with active set strategies.

CHAPTER 4

A GENERALIZATION: CYLINDRICAL INCLUSION

We study the problem of finding an ellipsoidal cylinder containing a finite set of points in $\mathbb{R}^n$, such that its cross-section with a k-dimensional subspace has minimum area. This is a generalization of the minimum-volume enclosing ellipsoid (MVEE) problem considered in Chapter 2.

The minimum-area enclosing ellipsoidal cylinder (MAEC) problem has also been widely studied, mainly because its dual is another optimal design problem in statistics, where now one is interested in estimating just k out of n parameters in a regression problem by choosing the design points optimally in some sense. See Fedorov [7], Silvey and Titterington [29], Atwood [3, 4], and Pukelsheim [22] for more details.

Our interest in this problem is mainly algorithmic: we study a first-order method based on the Frank-Wolfe method [8] with Wolfe's away steps [33], which was introduced in the context of optimal design by Atwood [4]. However, no detailed analysis of this method has been performed, and in certain cases it breaks down unless modified to keep an appropriate matrix positive definite.

We show that under reasonable conditions on the iterates produced by the algorithm, global complexity estimates and local convergence properties can be established. These conditions require that a certain principal submatrix of a positive semidefinite matrix produced by the algorithm remain positive definite.

We also provide a technique that allows the iterations to proceed when rank deficiency occurs; although we have no guarantee of convergence in this case,

the method appears to work in practice. Finally, some computational results for large random problems are given.

This chapter is organized as follows. In the next section, we state convex formulations of the MAEC problem and its dual. Although previous papers have included formulations with some convexity properties, they have not been fully convex. Section 4.2 describes the basic algorithm of Atwood [4]. We prove global and local convergence results in Section 4.3. The case of rank-deficiency of a critical submatrix is discussed in Section 4.5, and Section 4.6 contains the results of our computational study. We conclude in Section 4.7 with some final remarks.

4.1 Problem Formulation and Duality

In this section we provide convex formulations of the MAEC problem and its dual. Previous formulations, such as that in Silvey and Titterington [29], have not been convex in all the variables (see problem ($\mathcal{P}'$) below), although they have some convexity properties, and the dual problem involved a Schur complement rather than our simpler formulation ($\mathcal{D}$).

We also relate these formulations to earlier ones and prove duality results. The section ends by defining the notions of approximate optimality that will be used in our algorithms.

Suppose we are given a matrix $X = [x_1, x_2, \ldots, x_m] \in \mathbb{R}^{n\times m}$ whose columns, the points $x_1, \ldots, x_m$, span $\mathbb{R}^n$. Let $X = \begin{bmatrix} Z \\ Y \end{bmatrix}$ be a partition of X, where $Z \in \mathbb{R}^{(n-k)\times m}$ and $Y \in \mathbb{R}^{k\times m}$. If H' is a symmetric matrix of order k that is positive definite (we

write $H' \succ 0$) and E is a matrix of order $k \times (n-k)$, then the set

$$C(0; E; H') := \{[z; y] : z \in \mathbb{R}^{n-k}, y \in \mathbb{R}^k, (y + Ez)^T H'(y + Ez) \leq k\}$$

is a central (i.e., centered at the origin) ellipsoidal cylinder whose intersection with the subspace

$$\Pi := \{[z; y] \in \mathbb{R}^n : z \in \mathbb{R}^{n-k}, y \in \mathbb{R}^k, z = 0\}$$

has volume $(\det H')^{-1/2}$ times that of a Euclidean ball in $\mathbb{R}^k$ of radius $\sqrt{k}$. H' determines the shape of the cross-section and E the "directions of the axes" of the cylinder. Hence, finding a central ellipsoidal cylinder, which contains the columns of X and has minimum-volume (area) intersection with Π, amounts to solving

$$\begin{array}{rcl} \min_{H' \succ 0} \ \bar{f}(H', E) & := & -\ln \det H' \\ (\mathcal{P}') \quad (y_i + Ez_i)^T H'(y_i + Ez_i) & \leq & k, \quad i = 1, \ldots, m, \end{array}$$

where the variables are H' and E. This is called the minimum-area enclosing ellipsoidal cylinder (MAEC) problem.

This problem is nonconvex. The following lemma proves that it can be reformulated as a convex programming problem as follows:

$$\begin{array}{rcl} \min_{H_{YY} \succ 0} \ f(H) & := & -\ln \det H_{YY} \\ (\mathcal{P}) \qquad x_i^T H x_i & \leq & k, \quad i = 1, \ldots, m, \\ H & \succeq & 0, \end{array}$$

where the variable H is partitioned as $\begin{bmatrix} H_{ZZ} & H_{ZY} \\ H_{ZY}^T & H_{YY} \end{bmatrix}$ and $H_{YY} \in \mathbb{R}^{k \times k}$.

Lemma 4.1.1 *Problems* ($\mathcal{P}$) *and* ($\mathcal{P}'$) *are equivalent.*

Proof: To see this, consider any feasible solution H to $(\mathcal{P})$. Note that given $H_{YY} \succ 0$, $H \succeq 0$ holds iff $H_{ZZ} \geq H_{ZY}H_{YY}^{-1}H_{ZY}^T$. We can therefore assume that $H_{ZZ} = H_{ZY}H_{YY}^{-1}H_{ZY}^T$ without loss of generality, since replacing H_{ZZ} by the right-hand side will preserve feasibility and leave unchanged the objective value. Then

$$x_i^T H x_i \leq k$$

iff

$$y_i^T H_{YY} y_i + 2y_i^T H_{ZY}^T z_i + z_i^T H_{ZY} H_{YY}^{-1} H_{ZY}^T z_i \leq k.$$

If we let $E = H_{YY}^{-1}H_{ZY}^T$, the latter inequality holds iff

$$y_i^T H_{YY} y_i + 2y_i^T H_{YY} E z_i + z_i^T E^T H_{YY} E z_i \leq k$$

or

$$(y_i + Ez_i)^T H_{YY}(y_i + Ez_i) \leq k,$$

and we obtain a feasible solution to problem $(\mathcal{P}')$ with the same objective value. Conversely, given any feasible solution to $(\mathcal{P}')$, we can set $H_{YY} := H'$, $H_{ZY} := E^T H_{YY}$, and $H_{ZZ} := H_{ZY}H_{YY}^{-1}H_{ZY}^T$ to get a feasible solution to $(\mathcal{P})$ with the same objective value. Thus, problems $(\mathcal{P})$ and $(\mathcal{P}')$ are equivalent. □

When $k = n$, the MAEC problem reduces to the minimum-volume enclosing ellipsoid (MVEE) problem. We note that searching for a central ellipsoidal cylinder containing the columns of X is without loss of generality: if an arbitrary ellipsoidal cylinder is sought, it can be obtained by finding a central ellipsoidal cylinder in $\mathbb{R}^{n+1}$ containing the points $(1; x_i)$, $i = 1, \ldots, m$, with the same value of k.

Problem $(\mathcal{P})$ is convex with linear inequality constraints. After some simpli-

fication, its Lagrangian dual can be written as

$$
\begin{array}{lrcl}
\max_{u,K>0} & g(u,K) := \ln\det K & & \\
& XUX^T - \overline{K} := XUX^T - \begin{bmatrix} 0 & 0 \\ 0 & K \end{bmatrix} & \geq & 0, \\
(\mathcal{D}) & e^T u & = & 1, \\
& u & \geq & 0,
\end{array}
$$

where matrix U is a diagonal matrix with the components of u on its diagonal and e is a vector of ones in $\mathbb{R}^m$. We will see below Lemma 4.1.3 that $(\mathcal{D})$ is also equivalent to the statistical problem of finding a D_k-optimal design measure u on the columns of X, that is, one that maximizes the determinant of a $k \times k$ Schur complement of the Fisher information matrix $E[xx^T] = XUX^T$: see, e.g., Silvey and Titterington [29] and Fedorov [7].

We say H is feasible in $(\mathcal{P})$ if it is positive semidefinite and satisfies the constraints of $(\mathcal{P})$ and H_{YY} is positive definite; similarly, (u, K) is feasible in $(\mathcal{D})$ if it satisfies the constraints of $(\mathcal{D})$ and K is positive definite. We first establish weak duality:

Lemma 4.1.2 *$f(H) \geq g(u, K)$ for any H and (u, K) feasible in $(\mathcal{P})$ and $(\mathcal{D})$, respectively.*

Proof: Since H and $XUX^T - \overline{K}$ are positive semidefinite,

$$
0 \leq H \bullet (XUX^T - \overline{K}) = \sum_i u_i x_i^T H x_i - H \bullet \overline{K} \leq k e^T u - H_{YY} \bullet K. \tag{4.1.1}
$$

Hence we have

$$
\begin{array}{rcl}
-\ln\det H_{YY} - \ln\det K & = & -\ln\det H_{YY} K \\
& = & -k \ln(\Pi_{i=1}^k \lambda_i)^{1/k}
\end{array}
$$

$$
\begin{aligned}
&\geq -k \ln\left(\frac{\sum_{i=1}^{k} \lambda_i}{k}\right) \\
&\geq -k \ln\left(\frac{k}{k}\right) \\
&= 0,
\end{aligned}
$$

where the λ_i's are the positive eigenvalues of $H_{YY}K$ (or of ${H_{YY}}^{1/2}K{H_{YY}}^{1/2}$), and (4.1.1) shows their sum is at most k. □

Hence feasible solutions H and (u, K) are optimal if $f(H) = g(u, K)$. We next show that strong duality holds, so that this condition is necessary as well as sufficient.

Theorem 4.1.1 *There exist optimal solutions for problems* $(\mathcal{P})$ *and* $(\mathcal{D})$*. Furthermore, the following conditions, together with primal and dual feasibility, are necessary and sufficient for optimality in both* $(\mathcal{P})$ *and* $(\mathcal{D})$*:*

(a) $H \bullet (XUX^T - \overline{K}) = 0$,

(b) $u_i > 0$ *only if* $x_i^T H x_i = k$, *and*

(c) $H_{YY} = K^{-1}$.

Proof: Let H be a feasible solution for problem $(\mathcal{P})$. Summing up the linear constraints, we must have $\sum x_i^T H x_i = H \bullet XX^T \leq km$. Since $XX^T \succ 0$ and $km > 0$, $\{H \succeq 0 : H \bullet XX^T \leq km\}$ is a compact set. Hence the feasible region for problem $(\mathcal{P})$ is also a compact set (since it is the intersection of a compact set with a finite set of halfspaces). Moreover, $H = \epsilon I$ is feasible for $(\mathcal{P})$ for sufficiently small positive ϵ, so we can add the constraint that $\ln \det H_{YY} \geq k \ln \epsilon$, and the feasible region remains compact and has the same set of optimal solutions. Now the objective function is continuous on the compact feasible region, so an optimal

solution exists for problem $(\mathcal{P})$. Existence of an optimal solution for $(\mathcal{P})$ implies the existence of optimal solutions for $(\mathcal{P}')$ and also for $(\mathcal{D})$ as will be discussed later.

Sufficiency follows from the previous lemma, since the conditions imply equality in the weak duality inequality. In order to prove necessity, let $\tilde{H} = \begin{pmatrix} \tilde{H}_{ZZ} & \tilde{H}_{ZY} \\ \tilde{H}_{ZY}^T & \tilde{H}_{YY} \end{pmatrix}$ be an optimal solution for $(\mathcal{P})$, so that $(H' = \tilde{H}_{YY}, \tilde{E} = \tilde{H}_{YY}^{-1}\tilde{H}_{ZY}^T)$ is an optimal solution for the primal problem $(\mathcal{P}')$. Hence the Karush-Fritz-John conditions must hold for this solution, i.e., there exist nonnegative multipliers $\tilde{\nu} \in \mathbb{R}$ and $\tilde{u} \in \mathbb{R}^m$ with at least one of the multipliers nonzero and the following equalities hold:

$$-\tilde{\nu}H'^{-1} + \sum_{i=1}^{m} \tilde{u}_i(y_i + \tilde{E}z_i)(y_i + \tilde{E}z_i)^T = 0, \tag{4.1.2}$$

$$2\sum_{i=1}^{m} \tilde{u}_i H' y_i z_i^T + 2\sum_{i=1}^{m} \tilde{u}_i H' \tilde{E} z_i z_i^T = 0, \text{ and} \tag{4.1.3}$$

$$\tilde{u}_i((y_i + \tilde{E}z_i)^T H'(y_i + \tilde{E}z_i) - k) = 0, \forall i. \tag{4.1.4}$$

For one moment let's assume that $\tilde{\nu} = 0$. Then (4.1.2) becomes

$$\sum \tilde{u}_i(y_i + \tilde{E}z_i)(y_i + \tilde{E}z_i)^T = 0,$$

and hence (4.1.4) implies that $k \sum \tilde{u}_i = 0$. Since at least one of the $\tilde{u}_i$'s is positive, this is a contradiction and we must have $\tilde{\nu} > 0$. We can without loss of generality assume that $\tilde{\nu} = 1$, and conclude that any optimal solution $(H', \tilde{E})$ must satisfy

$$-H'^{-1} + \sum \tilde{u}_i(y_i + \tilde{E}z_i)(y_i + \tilde{E}z_i)^T = 0, \tag{4.1.5}$$

which together with (4.1.4) implies

$$\sum \tilde{u}_i = 1. \tag{4.1.6}$$

Since $H' \succ 0$, equation (4.1.3) can be written as $Y\tilde{U}Z^T + \tilde{E}Z\tilde{U}Z^T = 0$, and hence (4.1.5) becomes

$$\begin{aligned} H'^{-1} &= Y\tilde{U}Y^T + Y\tilde{U}Z^T\tilde{E}^T + \tilde{E}Z\tilde{U}Y^T + \tilde{E}Z\tilde{U}Z^T\tilde{E}^T \\ &= Y\tilde{U}Y^T - \tilde{E}Z\tilde{U}Z^T\tilde{E}^T. \end{aligned} \tag{4.1.7}$$

Let $\tilde{K} := Y\tilde{U}Y^T - \tilde{E}Z\tilde{U}Z^T\tilde{E}^T$. Then it is easy to check that $(\tilde{u}, \tilde{K})$ is a feasible solution for the dual problem, and that strong duality holds for the solution pair $\tilde{H}$ and $(\tilde{u}, \tilde{K})$. So strong duality must hold for any pair of optimal solutions H and (u, K). Hence conditions $(a) - (c)$ are both necessary and sufficient. □

Note that, by the strict convexity of the function $-\ln\det$, the H_{YY}-part of the optimal solution H of $(\mathcal{P})$ is unique, and hence by the theorem, so is the K part of the dual solution. However, there may be several optimal u's, and there may be several optimal E's for $(\mathcal{P}')$ (and H's for $(\mathcal{P})$). Indeed, as we show below, for any nonnegative u there may be several associated matrices E, but there is a unique associated $K = K(u)$, which is defined as is $\tilde{K}$ in the proof above.

Lemma 4.1.3 *Let u be dual feasible (nonnegative with components adding to one). Then:*

a) There exists E satisfying

$$EZUZ^T = -YUZ^T; \tag{4.1.8}$$

b) $K(u) := YUY^T - EZUZ^TE^T$ is independent of which E satisfying (4.1.8) is chosen; and

c) $XUX^T - \bar{K}$ is positive semidefinite iff $K \preceq K(u)$.

Proof: a) Suppose not. Then there is some q with $ZUZ^T q = 0$ but $YUZ^T q$ nonzero. But then

$$\begin{aligned} 0 &\leq \begin{pmatrix} q \\ p \end{pmatrix}^T \begin{bmatrix} ZUZ^T & ZUY^T \\ YUZ^T & YUY^T \end{bmatrix} \begin{pmatrix} q \\ p \end{pmatrix} \\ &= q^T ZUZ^T q + 2p^T YUZ^T q + p^T YUY^T p, \end{aligned}$$

which is negative for p a sufficiently small negative multiple of $YUZ^T q$, a contradiction.

b) Suppose E and E' satisfy (4.1.8). Then we have $(E - E')ZUZ^T = 0$ and hence

$$EZUZ^T E^T - E'ZUZ^T (E')^T = (E - E')ZUZ^T E' + EZUZ^T (E - E')^T = 0,$$

so that $K(u)$ is uniquely defined.

c) For any q and p, and any E satisfying the equation in (a),

$$\begin{aligned} \begin{pmatrix} q \\ p \end{pmatrix}^T (XUX^T - \bar{K}) \begin{pmatrix} q \\ p \end{pmatrix}^T &= \begin{pmatrix} q \\ p \end{pmatrix}^T \left(\begin{bmatrix} ZUZ^T & ZUY^T \\ YUZ^T & YUY^T - K \end{bmatrix} \right) \begin{pmatrix} q \\ p \end{pmatrix}^T \\ &= \begin{pmatrix} q \\ p \end{pmatrix}^T \begin{bmatrix} ZUZ^T & -ZUZ^T E^T \\ -EZUZ^T & EZUZ^T E \end{bmatrix} \begin{pmatrix} q \\ p \end{pmatrix}^T \\ &+ \begin{pmatrix} q \\ p \end{pmatrix}^T \begin{bmatrix} 0 & 0 \\ 0 & K(u) - K \end{bmatrix} \begin{pmatrix} q \\ p \end{pmatrix}^T \\ &= (q - E^T p)^T ZUZ^T (q - E^T p) + p^T (K(u) - K)p, \end{aligned}$$

and the result follows. □

The lemma implies that, for any dual feasible u, we can assume without loss of generality that $K = K(u)$, because the latter is feasible and yields at least as good an objective value. We therefore write $g(u)$ for $g(u, K(u))$, abusing notation slightly. It is easy to see that g is concave on the set of feasible u's. If u and u' are dual feasible, and $\bar{u} := (1 - \lambda)u + \lambda u'$ for $0 \leq \lambda \leq 1$, then $(\bar{u}, (1 - \lambda)K(u) + \lambda K(u'))$ is feasible in $(\mathcal{D})$, and hence $g(\bar{u}) \geq \ln\det((1 - \lambda)K(u) + \lambda K(u'))$. The result now follows from the concavity of $\ln\det$. The problem of maximizing $g(u, K(u))$ subject to u lying in the unit simplex is the formulation used earlier for the D_k-optimal design problem: see, e.g., Atwood [4] or Silvey and Titterington [29].

We have seen above two definitions of the axis matrix E: one from the primal problem $(\mathcal{P})$, $E = H_{YY}^{-1} H_{ZY}^T$, and one from a dual feasible solution as in (4.1.8). We now show that the first notion implies the second when feasible solutions H and u satisfy optimality condition (a): $H \bullet (XUX^T - \bar{K}) = 0$. (Here $\bar{K}$ is defined using $K = K(u)$.)

Indeed, if this equation holds, it also holds when H_{ZZ} is replaced by $E^T H_{YY} E$, where $E = H_{YY}^{-1} H_{ZY}^T$, which maintains positive semidefiniteness. But then

$$\begin{aligned} 0 &= \left(\begin{bmatrix} E^T \\ I \end{bmatrix} H_{YY} [E \quad I] \right) \bullet \begin{bmatrix} ZUZ^T & ZUY^T \\ YUZ^T & YUY^T - K \end{bmatrix} \\ &= H_{YY} \bullet \left([E \quad I] \begin{bmatrix} ZUZ^T & ZUY^T \\ YUZ^T & YUY^T - K \end{bmatrix} \begin{bmatrix} E^T \\ I \end{bmatrix} \right). \end{aligned}$$

Since H_{YY} is positive definite and the second matrix above is positive semidefinite, the latter must be zero. This then implies (e.g., by considering the positive semidefinite square root of $XUX^T - \bar{K}$) that

$$[E \quad I] \begin{bmatrix} ZUZ^T & ZUY^T \\ YUZ^T & YUY^T - K \end{bmatrix} = 0,$$

which implies (4.1.8).

Definition 4.1.1 *We call a dual feasible point* (u, K, E)*, i.e.,* $u \geq 0$*,* $e^T u = 1$*,* $YUZ^T = -E(ZUZ^T)$ *and* $K = YUY^T - EZUZ^T E^T \succ 0$*,* ϵ-primal feasible *if* $(y_i + Ez_i)^T K^{-1}(y_i + Ez_i) \leq (1+\epsilon)k$ *for all* i*, and say that it satisfies the* ϵ-approximate optimality conditions *or it is an* ϵ-approximate optimal solution *if moreover* $(y_i + Ez_i)^T K^{-1}(y_i + Ez_i) \geq (1-\epsilon)k$ *whenever* $u_i > 0$*.*

Lemma 4.1.4 *Given a dual feasible solution* (u, K, E) *which is* ϵ*-primal feasible, we have* $0 \leq g^* - g(u) \leq k \ln(1+\epsilon)$*, where* g^* *is the optimal objective function value of (D) and* $g(u) := g(u, K)$*.*

Proof: The ϵ-primal feasibility implies that $((1+\epsilon)^{-1}K^{-1}, E)$ is feasible for the primal problem $(\mathcal{P}')$. Then by weak duality we have

$$0 \leq g^* - g(u) \leq \bar{f}((1+\epsilon)^{-1}K^{-1}, E) - \ln\det K = k\ln(1+\epsilon).$$

□

4.2 The Algorithm

In the following two sections, we will assume that u and u_+ satisfy the following assumption:

Assumption 4.2.1 *The dual feasible variable* u *satisfies* $ZUZ^T \succ 0$ *where* $U := \mathrm{Diag}(u)$*.*

(We remark that the rank deficiency of ZU^*Z^T for the optimal solution u^* does not contradict this assumption's holding at all iterations, although some numerical instability might be expected.)

We will show that this assumption is not too restrictive by proposing a method for dealing with the rank-deficient case in Section 4.5. (Note that Atwood [4] suggests just reducing u_i to a very small positive value if making it zero during the algorithm would lead to rank deficiency.)

Using this assumption and the discussion in the previous section, we see that each dual feasible solution u is associated with a unique dual feasible variable $K = K(u) = YUY^T - YUZ^T(ZUZ^T)^{-1}ZUY^T$ and a unique axis matrix $E = E(u) = -(YUZ^T)(ZUZ^T)^{-1}$. As above, we will use $g(u)$ instead of $g(u, K)$. We can now motivate and describe our algorithm.

Let $\xi_i(u) := x_i^T(XUX^T)^{-1}x_i$, $\zeta_i(u) := z_i^T(ZUZ^T)^{-1}z_i$ and $\omega_i(u) := (y_i + Ez_i)^T K^{-1}(y_i + Ez_i)$. The following lemma will be useful in our analysis. It also shows that these quantities are all well-defined.

Lemma 4.2.1 *We have*

$$\xi_i(u) = \zeta_i(u) + \omega_i(u) \tag{4.2.9}$$

for all $i = 1, \ldots, m$.

Proof: Note first that

$$XUX^T = \begin{bmatrix} ZUZ^T & ZUY^T \\ YUZ^T & YUY^T \end{bmatrix} = \begin{bmatrix} I & 0 \\ -E & I \end{bmatrix} \begin{bmatrix} ZUZ^T & 0 \\ 0 & K \end{bmatrix} \begin{bmatrix} I & -E^T \\ 0 & I \end{bmatrix}. \tag{4.2.10}$$

(Since ZUZ^T (by our assumption) and K (since u is dual feasible) are positive definite, so is XUX^T, and thus ξ, ζ, and ω are all well-defined.) Hence we have

$$\begin{aligned}
\xi_i(u) &= x_i^T(XUX^T)^{-1}x_i \\
&= \begin{pmatrix} z_i^T & y_i^T \end{pmatrix}\begin{pmatrix} ZUZ^T & ZUY^T \\ YUZ^T & YUY^T \end{pmatrix}^{-1}\begin{pmatrix} z_i \\ y_i \end{pmatrix} \\
&= \begin{pmatrix} z_i^T & y_i^T \end{pmatrix}\begin{pmatrix} I & E^T \\ 0 & I \end{pmatrix}\begin{pmatrix} (ZUZ^T)^{-1} & 0 \\ 0 & K^{-1} \end{pmatrix}\begin{pmatrix} I & 0 \\ E & I \end{pmatrix}\begin{pmatrix} z_i \\ y_i \end{pmatrix} \\
&= \begin{pmatrix} z_i^T & y_i^T + z_i^T E^T \end{pmatrix}\begin{pmatrix} (ZUZ^T)^{-1} & 0 \\ 0 & K^{-1} \end{pmatrix}\begin{pmatrix} z_i \\ y_i + Ez_i \end{pmatrix} \\
&= z_i^T(ZUZ^T)^{-1}z_i + (y_i + Ez_i)^T K^{-1}(y_i + Ez_i) \\
&= \zeta_i(u) + \omega_i(u). \quad \square
\end{aligned}$$

(Those readers familiar with the MVEE problem will notice that it is a trivial special case in which we have $k = n$, $\omega(u) = \xi(u)$ and $\zeta(u) = 0$.)

Now note that assuming $ZUZ^T \succ 0$, from (4.2.10) we get $\det K = \frac{\det XUX^T}{\det ZUZ^T}$. Hence

$$\begin{aligned}
\nabla g(u) &= \nabla(\ln\det XUX^T - \ln\det ZUZ^T) \\
&= \xi(u) - \zeta(u) = \omega(u). \qquad (4.2.11)
\end{aligned}$$

Suppose we make an update of the form

$$u_+ := (1-\tau)u + \tau e_j, \qquad (4.2.12)$$

where $e_j = (0, \ldots, 1, \ldots, 0)^T$ is the j^{th} unit vector. Then

$$ZU_+Z^T = (1-\tau)\left(ZUZ^T + \frac{\tau}{1-\tau}z_jz_j^T\right) = (1-\tau)(ZUZ^T + \lambda z_jz_j^T),$$

and using the Sherman-Morrison formula, we have

$$(ZU_+Z^T)^{-1} = \frac{1}{1-\tau}\left((ZUZ^T)^{-1} - \frac{\lambda}{1+\lambda\zeta_j(u)}(ZUZ^T)^{-1}z_jz_j^T(ZUZ^T)^{-1}\right)$$

$$= \frac{1}{1-\tau}\left((ZUZ^T)^{-1} - \mu(ZUZ^T)^{-1}z_j z_j^T(ZUZ^T)^{-1}\right), \tag{4.2.13}$$

where $\mu = \frac{\lambda}{1+\lambda\zeta_j(u)}$. Next,

$$YU_+Z^T = (1-\tau)(YUZ^T + \lambda y_j z_j^T), \tag{4.2.14}$$

and so

$$\begin{aligned}
E_+ &= -(YU_+Z^T)(ZU_+Z^T)^{-1} \\
&= -(YUZ^T + \lambda y_j z_j^T)\Big((ZUZ^T)^{-1} - \mu(ZUZ^T)^{-1}z_j z_j^T(ZUZ^T)^{-1}\Big) \\
&= E - \lambda y_j z_j^T(ZUZ^T)^{-1} - \mu E z_j z_j^T(ZUZ^T)^{-1} + \mu\lambda\zeta_j(u) y_j z_j^T(ZUZ^T)^{-1} \\
&= E - \Big(\lambda(1-\mu\zeta_j(u))y_j + \mu E z_j\Big)z_j^T(ZUZ^T)^{-1} \\
&= E - \mu(y_j + Ez_j)z_j^T(ZUZ^T)^{-1}.
\end{aligned} \tag{4.2.15}$$

Finally,

$$YU_+Y^T = (1-\tau)(YUY^T + \lambda y_j y_j^T), \tag{4.2.16}$$

so

$$\begin{aligned}
K_+ &= YU_+Y^T + YU_+Z^T E_+^T \\
&= (1-\tau)\Big(YUY^T + \lambda y_j y_j^T + (YUZ^T + \lambda y_j z_j^T)(E^T - \mu(ZUZ^T)^{-1}z_j(y_j + Ez_j)^T)\Big) \\
&= (1-\tau)\Big(K + \lambda y_j(y_j + Ez_j)^T + \mu E z_j(y_j + Ez_j)^T - \lambda\mu\zeta_j(u) y_j(y_j + Ez_j)^T\Big) \\
&= (1-\tau)\Big(K + (\lambda(1-\mu\zeta_j(u))y_j + \mu E z_j)(y_j + Ez_j)^T\Big) \\
&= (1-\tau)\Big(K + \mu(y_j + Ez_j)(y_j + Ez_j)^T\Big).
\end{aligned} \tag{4.2.17}$$

We have

$$\ln\det K_+ = \ln\det K - k\ln(1+\lambda) + \ln(1+\mu\omega_j(u)), \tag{4.2.18}$$

whose derivative is

$$\frac{\partial \ln\det K_+}{\partial\lambda} = \frac{-k}{1+\lambda} + \frac{1}{1+\frac{\lambda\omega_j(u)}{1+\lambda\zeta_j(u)}}\left(\frac{\omega_j(u)}{1+\lambda\zeta_j(u)} - \frac{\lambda\zeta_j(u)\omega_j(u)}{(1+\lambda\zeta_j(u))^2}\right)$$

$$
\begin{aligned}
&= \frac{-k}{1+\lambda} + \frac{\omega_j(u)}{(1+\lambda(\zeta_j(u)+\omega_j(u)))(1+\lambda\zeta_j(u))} \\
&= \frac{-k}{(1+\lambda)(1+\lambda\zeta_j(u))(1+\lambda\xi_j(u))}\Big(a\lambda^2 - 2b\lambda + c\Big),
\end{aligned}
\tag{4.2.19}
$$

where $a := \zeta_j(u)\xi_j(u) \geq 0$, $b := -\zeta_j(u) - \frac{\omega_j(u)}{2} + \frac{\omega_j(u)}{2k} \leq 0$, and $c := 1 - \frac{\omega_j(u)}{k}$. The multiplier on the left of the quadratic is negative for all feasible values of λ. We can find the best step size τ (or λ) by investigating the roots of the quadratic equation $a\lambda^2 - 2b\lambda + c = 0$ and the boundary condition ($\lambda \geq -u_j$) arising from the nonnegativity of the dual feasible solution. Having obtained the optimal λ as below, we set $\tau = \frac{\lambda}{1+\lambda}$. Note that if λ is set to $-u_j$, τ is $\frac{-u_j}{1-u_j}$ and $(u_+)_j$ is zero.

- If $c > 0$ (i.e., $\omega_j(u) < k$), the derivative is negative for $\lambda = 0$, so we want to decrease λ.
  - If $a = 0$ and $b < 0$, set $\lambda = \max\{\frac{c}{2b}, -u_j\}$; if $a = b = 0$ set $\lambda = -u_j$;
  - if $a > 0$ and $-b \leq \sqrt{ac}$, then the derivative is negative for all $\lambda < 0$ (possibly zero at one point), so set $\lambda = -u_j$ and hence $\tau = \frac{-u_j}{1-u_j}$ so that $(u_+)_j = 0$; and
  - if $a > 0$ and $-b > \sqrt{ac}$, then the derivative is negative up to the root of the quadratic, so set $\lambda = \max\{\frac{c}{b - \sqrt{b^2-ac}}, -u_j\}$.

- If $c < 0$ (i.e., $\omega_j(u) > k$), the derivative is positive for $\lambda = 0$, so we want to increase λ.
  - If $a = 0$ and $b < 0$, set $\lambda = \frac{c}{2b}$; if $a = b = 0$, set $\lambda = \infty$ and $\tau = 1$ (this can only happen if $k = 1$); and
  - if $a > 0$, so $ac < 0$ and hence the quadratic has a positive root, set $\lambda = \frac{c}{b - \sqrt{b^2-ac}}$.

Once we determine the step size τ, we can find $\omega(u_+)$ and $\zeta(u_+)$ cheaply (in $O(mn)$ work) from

$$\begin{aligned} K_+^{-1} &= \frac{1}{1-\tau}(K^{-1} - \frac{\lambda}{1+\lambda\xi_j(u)}K^{-1}(y_j + Ez_j)(y_j + Ez_j)^T K^{-1}) \\ &= \frac{1}{1-\tau}(K^{-1} - \eta K^{-1}(y_j + Ez_j)(y_j + Ez_j)^T K^{-1}), \end{aligned} \tag{4.2.20}$$

where $\eta := \frac{\lambda}{1+\lambda\xi_j(u)}$. Then

$$\begin{aligned} \omega_i(u_+) &= (y_i + E_+ z_i)^T K_+^{-1}(y_i + E_+ z_i) \\ &= \frac{1}{1-\tau}(y_i + (E - \mu(y_j + Ez_j)z_j^T(ZUZ^T)^{-1})z_i)^T(K^{-1} - \eta K^{-1}(y_j + Ez_j) \\ &\quad \ldots (y_j + Ez_j)^T K^{-1})(y_i + (E - \mu(y_j + Ez_j)z_j^T(ZUZ^T)^{-1})z_i) \\ &= \frac{1}{1-\tau}(y_i + Ez_i - \mu\zeta_{ij}(u)(y_j + Ez_j))^T(K^{-1} - \eta K^{-1}(y_j + Ez_j)(y_j + Ez_j)^T K^{-1}) \\ &\quad \ldots (y_i + Ez_i - \mu\zeta_{ij}(u)(y_j + Ez_j)) \\ &= \frac{1}{1-\tau}(y_i + Ez_i - \mu\zeta_{ij}(u)(y_j + Ez_j))^T(K^{-1}(y_i + Ez_i) - \eta\xi_{ij}(u)K^{-1}(y_j + Ez_j)) \\ &= \frac{1}{1-\tau}(\omega_i(u) - \eta\omega_{ij}^2(u) - 2\eta\zeta_{ij}(u)\omega_{ij}(u) + \eta\mu\omega_j(u)\zeta_{ij}^2(u)), \end{aligned}$$

and

$$\begin{aligned} \zeta_i(u_+) &= z_i^T(ZU_+Z^T)^{-1}z_i \\ &= \frac{1}{1-\tau}\left(z_i^T\left((ZUZ^T)^{-1} - \mu(ZUZ^T)^{-1}z_j z_j^T(ZUZ^T)^{-1}\right)z_i\right) \\ &= \frac{1}{1-\tau}\left(\zeta_i(u) - \mu\zeta_{ij}(u)^2\right), \end{aligned} \tag{4.2.21}$$

where $\xi_{ij}(u) := x_i^T(XUX^T)^{-1}x_j$, $\zeta_{ij}(u) := z_i^T(ZUZ^T)^{-1}z_j$ and $\omega_{ij}(u) := (y_i + Ez_i)^T K^{-1}(y_j + Ez_j)$. Note that the following lemma has been used in the previous derivations and can be proved with arguments similar to those used in the proof of Lemma 4.2.1.

Lemma 4.2.2 *$\xi_{ij}(u) = \zeta_{ij}(u) + \omega_{ij}(u)$, for all i and j.*

To calculate these quantities, we need only compute inner products once we have $(XUX^T)^{-1}x_j$, $(ZUZ^T)^{-1}z_j$, and $K^{-1}(y_j + EZ_j)$, all of which are easy to obtain if we maintain a Cholesky factor $L = \begin{bmatrix} L_{ZZ} & 0 \\ L_{YZ} & L_{YY} \end{bmatrix}$ of XUX^T, since then L_{ZZ} is a Cholesky factor of ZUZ^T and L_{YY} of K.

Note that (4.2.21) gives in particular

$$\zeta_j(u_+) = \frac{(1 - \mu\zeta_j(u))\zeta_j(u)}{1 - \tau} = \frac{(1 + \lambda)\zeta_j(u)}{1 + \lambda\zeta_j(u)} \tag{4.2.22}$$

and it is easy to see that similar arguments lead to

$$\xi_j(u_+) = \frac{(1 + \lambda)\xi_j(u)}{1 + \lambda\xi_j(u)}. \tag{4.2.23}$$

Now, let us consider the following Frank-Wolfe type algorithm:

Algorithm 4.2.1

Input: $X \in \mathbb{R}^{n\times m}$, $k \in \{1,\ldots,n\}$ *and* $\epsilon > 0$.

Step 0. *Initialize* $u = e/m$. *Compute* K, E, $\omega(u)$, *and* $\zeta(u)$.

Step 1. *Find* $s := \arg\max\{\omega_i(u) - k\}$, $t := \arg\min\{\omega_i(u) - k : u_i > 0\}$.

If $\omega_s(u) - k \leq \epsilon k$ *and* $\omega_t(u) - k \geq -\epsilon k$,

STOP: *u is an* ϵ*-approximate optimal solution.*

Else,

if $\omega_s(u) - k > k - \omega_t(u)$, ***go to*** *Step 2;*

else, go to *Step 3.*

Step 2. *Replace u by* $u_+ := (1-\tau)u + \tau e_s$, *where* $\tau > 0$ *is chosen as in Section 4.2 to maximize g.* ***Go to*** *step 4.*

Step 3. *Replace u by* $u_+ := (1-\tau)u + \tau e_t$, *where now* τ *is chosen from negative values to maximize g subject to* u_+ *remaining feasible.*

Step 4. *Update* K, E, $\omega(u)$, *and* $\zeta(u)$. ***Go to*** *Step 1.*

The algorithm starts with a feasible dual solution $u = e/m$ and stops when an ϵ-approximate optimal solution is found. Theoretically, we are able to prove that the initial solution described above is not far from the optimal one in some sense. In practice, using the procedure described by [18] provides better initial solutions but without any performance guarantee. Until an ϵ-approximate optimal solution is found, a Frank-Wolfe type step such as (4.2.12) is taken at each iteration (Steps 2, 3, and 4 above). At each iteration one of the m vertices of the unit simplex is chosen and the new solution is obtained by either moving towards (called an increase or add step) or away from (called a decrease or drop step) this vertex. There are alternative ways to choose the vertex. Algorithm

4.2.1 picks one of the two vertices which maximize or minimize a linear approximation to g. (Choosing one of these two vertices has both theoretical and practical advantages as we will discuss later.) Once this decision is made, the best step size is found by a line search on the line segment joining our current iterate to the optimal solution of the linearized problem. Alternatively, we can calculate the improvement in the objective function value g for each vertex and choose the best one. This version will be referred as Algorithm 4.2.1-ALL. We will discuss the practical implications of these decisions in Section 4.6.

4.3 Convergence Analysis of the Algorithm

We discuss the global convergence properties of the algorithm in this section. The following lemma will be very useful in our analysis.

Lemma 4.3.1 *For any dual feasible solution u satisfying Assumption 4.2.1, we have*

$$\sum_{i=1}^{m} u_i \omega_i(u) = k. \tag{4.3.24}$$

Proof: Using (4.2.9)

$$\begin{aligned}
\sum_{i=1}^{m} u_i \omega_i(u) &= \sum_{i=1}^{m} u_i \left(\xi_i(u) - \zeta_i(u)\right) \\
&= \sum_{i=1}^{m} u_i \xi_i(u) - \sum_{i=1}^{m} u_i \zeta_i(u) \\
&= \sum_{i=1}^{m} u_i x_i^T (XUX^T)^{-1} x_i - \sum_{i=1}^{m} u_i z_i^T (ZUZ^T)^{-1} z_i \\
&= XUX^T \bullet (XUX^T)^{-1} - ZUZ^T \bullet (ZUZ^T)^{-1} \\
&= n - (n-k) = k. \quad \square
\end{aligned}$$

If we look closely at the algorithm described in the previous section, we can identify four different types of iterations. Let u^l be the dual feasible solution at hand at iteration number l, e_{j_l} be the vertex that we use in our update and τ_l be the step size associated with this update. We refer to iteration l as

- an *increase step* if $u^l_{j_l} > 0$ and $\tau_l > 0$,
- an *add step* if $u^l_{j_l} = 0$ and $\tau_l > 0$,
- a *decrease step* if $u^l_{j_l} > 0$ and $\frac{-u^l_{j_l}}{1-u^l_{j_l}} < \tau_l < 0$, and
- a *drop step* if $u^l_{j_l} > 0$ and $\tau_l = \frac{-u^l_{j_l}}{1-u^l_{j_l}}$.

Note that after a drop step we have $u^{l+1}_{j_l} = 0$. In a drop step, we may not be able to improve the objective function as much as we desire. Fortunately, the number of drop steps is bounded above by the number of add steps plus a constant (m, the number of positive components of the initial solution), and hence studying only the first three types of steps will be enough to obtain convergence results.

The following lemma gives a bound on the improvement obtained at each iteration, assuming that all the quantities $\xi_l(u^l)$ are uniformly bounded by some positive C at all iterations. The global convergence estimate then depends on this constant C. In practice it appears that these quantities are bounded by a reasonable constant; unfortunately, we have not been able to establish global convergence without this assumption.

Lemma 4.3.2 *Let u^l be the dual solution at the l^{th} iteration of the algorithm and $g_l := g(u^l)$. Assume that u^l satisfies Assumption 4.2.1 for all $l = 1, 2, \ldots$ Let ϵ_l be the smallest number such that u^l satisfies the ϵ_l-approximate optimality conditions. If*

for some positive constant C

$$\xi(u^l) \leq Ce \ for \ all \ l = 1, 2, \ldots,$$

then we have

$$g_0 > -\infty, \ \epsilon_0 \leq m - 1, \tag{4.3.25}$$

$$\Delta_l = g_{l+1} - g_l \geq \ln(1 + \hat{c}\epsilon_l^2), \ l = 0, 1, \ldots, \tag{4.3.26}$$

whenever l is not a drop step, where $\hat{c} = \hat{c}(C) > 0$*, and*

$$\delta_l = g^* - g_l \leq k \ln(1 + \epsilon_l), \ l = 0, 1, \ldots. \tag{4.3.27}$$

Proof: From Lemma 4.3.1 we have

$$\frac{1}{m} \sum_{i=1}^{m} \omega_i(u^0) = k.$$

By definition of ϵ_0,

$$k(1 + \epsilon_0) = \max\{\omega_i(u^0) | i = 1, .., m\} \leq km,$$

or

$$k(1 - \epsilon_0) = \min\{\omega_i(u^0) | i = 1, .., m\} \geq 0,$$

which implies $\epsilon_0 \leq \max\{1, m - 1\}$, and hence (4.3.25) holds.

For simplicity of notation, let $j := j_l$, $\tau = \tau_l$, and $\lambda = \lambda_l$. In order to prove (4.3.26), we can use (4.2.22) and (4.2.23) to get

$$\begin{aligned} \omega_j(u^{l+1}) &= \xi_j(u^{l+1}) - \zeta_j(u^{l+1}) \\ &= \frac{(1 + \lambda)\xi_j(u^l)}{1 + \lambda\xi_j(u^l)} - \frac{(1 + \lambda)\zeta_j(u^l)}{1 + \lambda\zeta_j(u^l)} \\ &= \frac{(1 + \lambda)\omega_j(u^l)}{(1 + \lambda\xi_j(u^l))(1 + \lambda\zeta_j(u^l))}. \end{aligned} \tag{4.3.28}$$

Now, let's assume that iteration l is an add or increase step, i.e., $j = \arg\max_i \omega_i(u^l)$ and $\omega_j(u^l) = (1+\epsilon_l)k$, so that $\hat{\lambda}$ (the optimal step length) is positive. Then for any λ such that $0 \le \lambda \le \hat{\lambda}$,

$$\begin{aligned} k &\le \frac{(1+\lambda)\omega_j(u^l)}{(1+\lambda\xi_j(u^l))(1+\lambda\zeta_j(u^l))} \\ &\le \frac{(1+\lambda)\omega_j(u^l)}{(1+\lambda_l\zeta_j(u^l))^2}. \end{aligned} \tag{4.3.29}$$

Therefore we have

$$(1+\lambda\zeta_j(u^l))^2 \le (1+\lambda)\omega_j(u^l)/k = (1+\lambda)(1+\epsilon_l),$$

which gives

$$1+\lambda\zeta_j(u^l) \le (1+\lambda)\sqrt{1+\epsilon_l}. \tag{4.3.30}$$

Note that since ϵ_l is bounded above ($(1+\epsilon_l)k = \omega_j(u^l) \le \xi(u^l) \le C$) we can find a constant $c_1 > 0$ such that $\sqrt{1+\epsilon_l} \ge 1 + c_1\epsilon_l$, and using this and (4.2.18), we obtain (for $\lambda < 1/k$)

$$\begin{aligned} g_{l+1} - g_l &= \ln\{(1+\lambda)^{-k}\left(1 + \frac{\lambda\omega_j(u^l)}{1+\lambda\zeta_j(u^l)}\right)\} \\ &\ge \ln\{(1-k\lambda)\left(1 + \frac{\lambda k(1+\epsilon_l)}{(1+\lambda)\sqrt{1+\epsilon_l}}\right)\} \\ &= \ln\{(1-k\lambda)\left(1 + \frac{\lambda k\sqrt{1+\epsilon_l}}{1+\lambda}\right)\} \\ &\ge \ln\{(1-k\lambda)\left(1 + \frac{\lambda k(1+c_1\epsilon_l)}{1+\lambda}\right)\}. \end{aligned} \tag{4.3.31}$$

Now choose λ' and c_0 positive and small enough such that for all $0 \le \lambda \le \lambda'$ the following inequality holds:

$$(1-k\lambda)(1 + \frac{\lambda k(1+c_1\epsilon_l)}{1+\lambda}) \ge 1 + \lambda c_0\epsilon_l. \tag{4.3.32}$$

For $\lambda = \min\{\hat{\lambda}, \lambda'\}$, this leads to the following bound:

$$g_{l+1} - g_l \ge \ln(1 + \lambda c_0\epsilon_l). \tag{4.3.33}$$

Note that (4.3.32) can be rewritten as

$$\begin{aligned}(1-k\lambda)(1+\lambda+\lambda k+\lambda k c_1\epsilon_l) &\geq (1+\lambda c_0\epsilon_l)(1+\lambda), \text{ or}\\ \lambda k c_1\epsilon_l - k\lambda^2 - k^2\lambda^2 - k^2\lambda^2 c_1\epsilon_l &\geq \lambda c_0\epsilon_l + \lambda^2 c_0\epsilon_l, \text{ or}\\ \lambda^2(c_0\epsilon_l + k + k^2 + k^2 c_1\epsilon_l) + \lambda(c_0\epsilon_l - k c_1\epsilon_l) &\leq 0. \end{aligned} \tag{4.3.34}$$

We can assume that $\lambda' = \frac{-c_0\epsilon_l + kc_1\epsilon_l}{c_0\epsilon_l + k + k^2 + k^2 c_1\epsilon_l}$ and choose $c_0 < kc_1$, and have the following bound for some constant c_2:

$$\lambda' \geq c_2\epsilon_l > 0. \tag{4.3.35}$$

On the other hand, since $\hat{\lambda}$ is the optimal step length we have

$$\begin{aligned} k = \omega_j(u^{l+1}) &= \frac{(1+\hat{\lambda})\omega_j(u^l)}{(1+\hat{\lambda}\xi_j(u^l))(1+\hat{\lambda}\zeta_j(u^l))}\\ &\geq \frac{(1+\hat{\lambda})\omega_j(u^l)}{(1+\hat{\lambda}\xi_j(u^l))^2}. \end{aligned}$$

Therefore we have

$$(1+\hat{\lambda}\xi_j(u^l))^2 \geq \omega_j(u^l)/k = 1+\epsilon_l,$$

which leads to

$$1+\hat{\lambda}\xi_j(u^l) \geq 1 + c_1\epsilon_l$$

and hence

$$\hat{\lambda} \geq \frac{c_1}{\xi_j(u^l)}\epsilon_l \geq \frac{c_1}{C}\epsilon_l = c_3\epsilon_l. \tag{4.3.36}$$

Equations (4.3.35) and (4.3.36) can be combined to obtain

$$\lambda = \min\{\lambda', \hat{\lambda}\} \geq c_4\epsilon_l > 0, \tag{4.3.37}$$

where $c_4 = \min\{c_2, c_3\}$. This inequality together with (4.3.33) gives (4.3.26) whenever iteration l is an add or increase step.

The proof of the case where iteration l is a decrease step is very similar: Let's assume that iteration l is a decrease step, i.e., $j = \arg\min\{\omega_i(u^l) : u_i^l > 0\}$, $\omega_j(u^l) = (1-\epsilon_l)k$ and $\hat{\lambda}$ (the optimal step length) satisfies $-u_j^l < \hat{\lambda} < 0$. For any λ such that $0 \geq \lambda \geq \hat{\lambda}$, we have

$$k \geq \frac{(1+\lambda)\omega_j(u_l)}{(1+\lambda\zeta_j(u_l))^2}. \tag{4.3.38}$$

Therefore we get

$$(1+\lambda\zeta_j(u_l))^2 \geq (1+\lambda)\omega_j(u_l)/k \geq (1+\lambda)^2(1-\epsilon_l),$$

which in turn gives

$$1+\lambda\zeta_j(u_l) \geq (1+\lambda)\sqrt{1-\epsilon_l}. \tag{4.3.39}$$

Hence we have

$$\begin{aligned} g_{l+1} - g_l &= \ln(1+\lambda)^{-k}(1+\frac{\lambda_l\omega_j(u_l)}{1+\lambda\zeta_j(u_l)}) \\ &\geq \ln(1-k\lambda)(1+\frac{k\lambda(1-\epsilon_l)}{(1+\lambda)\sqrt{1-\epsilon_l}}) \\ &\geq \ln(1-k\lambda)(1+\frac{k\lambda\sqrt{1-\epsilon_l}}{(1+\lambda)}) \\ &\geq \ln(1-k\lambda)(1+\frac{k\lambda(1-\epsilon_l/2)}{(1+\lambda)}). \end{aligned} \tag{4.3.40}$$

Now choose $\lambda' < 0$ large enough such that for all $0 \geq \lambda \geq \lambda'$ the following inequality holds:

$$(1-k\lambda)(1+\frac{k\lambda(1-\epsilon_l/2)}{(1+\lambda)}) \geq 1-\frac{\lambda k\epsilon_l}{3}. \tag{4.3.41}$$

For $\lambda = \max(\hat{\lambda}, \lambda')$, we have the following bound:

$$g_{l+1} - g_l \geq \ln\left(1-\tfrac{\lambda_l k\epsilon_l}{3}\right). \tag{4.3.42}$$

Note that (4.3.41) can be rewritten as

$$\begin{aligned} (1-k\lambda)(1+\lambda+\lambda k-\lambda k\epsilon_l/2) &\geq 1+\lambda-\lambda k\epsilon_l/3-\lambda^2 k\epsilon_l/3, \text{or} \\ -\lambda k\epsilon_l/2-\lambda^2 k-\lambda^2 k^2+\lambda^2 k^2\epsilon_l/2 &\geq -\lambda k\epsilon_l/3-\lambda^2 k\epsilon_l/3, \text{ or} \\ \lambda^2(k^2+k-k\epsilon_l/3-k^2\epsilon_l/2)+\lambda(k\epsilon_l/2-k\epsilon_l/3) &\leq 0. \end{aligned} \tag{4.3.43}$$

Hence we can choose $\lambda' = \frac{\epsilon_l/3-\epsilon_l/2}{k+1-\epsilon_l/3-k\epsilon_l/2}$, and we have

$$|\lambda'| \geq c_5\epsilon_l. \tag{4.3.44}$$

Since $\hat{\lambda}$ is the optimal step length we have

$$\begin{aligned} k = \omega_j(u_{l+1}) &= \frac{(1+\hat{\lambda})\omega_j(u_l)}{(1+\hat{\lambda}\xi_j(u_l))(1+\hat{\lambda}\zeta_j(u_l))} \\ &\leq \frac{(1+\hat{\lambda})\omega_j(u_l)}{(1+\hat{\lambda}\xi_j(u_l))^2}, \end{aligned}$$

which gives

$$(1+\hat{\lambda}\xi_j(u_l))^2 \leq \omega_j(u_l)/k = 1-\epsilon_l. \tag{4.3.45}$$

From this we get $1+\hat{\lambda}\xi_j(u_l) \leq \sqrt{1-\epsilon_l} \leq 1-\epsilon_l/2$. Hence $\hat{\lambda} \leq \frac{-\epsilon_l}{2\xi_j(u_l)} \leq \frac{-\epsilon_l}{2C}$ and

$$|\hat{\lambda}| \geq c_6\epsilon_l. \tag{4.3.46}$$

Equations (4.3.44) and (4.3.46) lead to

$$|\lambda| \geq \min\{c_5, c_6\}\epsilon_l = c_7\epsilon_l, \tag{4.3.47}$$

which combined with (4.3.42) gives (4.3.26) for decrease steps.

Finally, (4.3.27) follows from Lemma 4.1.4. □

Lemma 4.3.3 *Let $\epsilon \in (0,1)$. Under the assumptions of Lemma 4.3.2, Algorithm 4.2.1 obtains an ϵ-approximate optimal solution in at most*

$$L(\epsilon) = O(m + k(\epsilon^{-1} + \ln k + \ln\ln m)) \tag{4.3.48}$$

iterations.

Note that the "big oh" here and in Theorem 4.3.1 below contains constants that depend on C in Lemma 4.3.2, and also relies on Assumption 4.2.1.

Proof: We will first show that

$$L(1) := \min\{l : \epsilon_l \leq 1\} = O(k(\ln k + \ln\ln m)). \tag{4.3.49}$$

First note that as long as $\epsilon_l \geq 1$, the l^{th} iteration of Algorithm 4.2.1 can only be an add or increase step. Furthermore, $\epsilon_l \geq 1$, (4.3.26), and (4.3.27) imply that

$$\delta_l - \delta_{l+1} = \Delta_l \geq \ln(1 + \hat{c}\epsilon_l^2) \geq \ln(1 + \hat{c}\epsilon_l) \geq \tilde{c}\ln(1 + \epsilon_l) \geq \frac{\tilde{c}}{k}\delta_l$$

for some $\tilde{c} = \tilde{c}(C) = \min\{1, \ln_2(1 + \hat{c})\} > 0$. Hence

$$\delta_{l+1} \leq (1 - \frac{\tilde{c}}{k})\delta_l,$$

which implies

$$\delta_l \leq \delta_0(1 - \frac{\tilde{c}}{k})^l \leq \delta_0 e^{-\tilde{c}l/k}. \tag{4.3.50}$$

From (4.3.25) and (4.3.27), we have

$$\delta_0 \leq k\ln(1 + \epsilon_0) \leq k\ln m. \tag{4.3.51}$$

$\epsilon_l \geq 1$ also implies that

$$\delta_l \geq \Delta_l \geq \bar{c} > 0, \tag{4.3.52}$$

where $\bar{c} = \ln(1 + \hat{c})$. Hence from (4.3.50), (4.3.51) and (4.3.52), we get

$$L(1) \leq \frac{k}{\tilde{c}}\ln\frac{k\ln m}{\bar{c}} \leq O(k(\ln k + \ln\ln m)).$$

Now assume that $\epsilon_l \leq 1$, and let $h(\epsilon_l)$ be the number of add, increase, and decrease steps required to obtain an $\epsilon_l/2$-approximate optimal solution starting with an ϵ_l-approximate optimal solution. As long as $\epsilon_{l+h} \geq \epsilon_l/2$ and $l + h$ is an add, increase, or decrease step, we also have

$$\Delta_{l+h} \geq \ln(1 + \hat{c}(\epsilon_l/2)^2) \geq \frac{\hat{c}_1}{8}\epsilon_l^2 > 0,$$

where $\hat{c}_1 := \min\{4, \hat{c}\}$. Since we have $\delta_l \geq \delta_{l+1} \geq \ldots$, and

$$\delta_l \leq k \ln(1 + \epsilon_l) \leq k\epsilon_l,$$

we obtain the following bound:

$$h(\epsilon_l) \leq \frac{\delta_l}{\hat{c}_1 \epsilon_l^2/8} \leq \frac{8k}{\hat{c}_1 \epsilon_l}. \tag{4.3.53}$$

Applying this argument repeatedly, we conclude that we need at most

$$\begin{aligned} \mathcal{H}(\epsilon) &= h(\epsilon_l) + h(\epsilon_l/2) + \ldots + h(\epsilon_l/2^{\lceil \ln \epsilon_l/\epsilon \rceil - 1}) \\ &\leq \frac{8k}{\hat{c}_1}\left[\frac{1}{\epsilon_l} + \frac{2}{\epsilon_l} + \ldots + \frac{2^{\lceil \ln \epsilon_l/\epsilon \rceil - 1}}{\epsilon_l}\right] \leq \frac{16k}{\hat{c}_1 \epsilon} = O\left(\frac{k}{\epsilon}\right) \end{aligned} \tag{4.3.54}$$

add, increase, and decrease iterations to obtain an ϵ-approximate optimal solution starting with an ϵ_l-approximate optimal solution where $\epsilon_l \in (0, 1]$. Since the number of drop steps is bounded above by the number of add steps plus m (the number of positive components of the initial solution u^0), (4.3.48) is immediate. □

We can implement this algorithm using rank-one update formulae so that each iteration takes $O(nm)$ arithmetic operations and comparisons. Hence the following theorem follows from Lemma 4.3.3.

Theorem 4.3.1 *Let $\epsilon \in (0, 1)$. Under the assumptions of Lemma 4.3.2, Algorithm 4.2.1 finds an ϵ–approximate optimal solution to the MAEC problem in*

$$N(\epsilon) = O(knm(\epsilon^{-1} + \ln k + \ln \ln m) + nm^2) \tag{4.3.55}$$

arithmetic operations and comparisons.

In cases where $m \gg n$ and n/k is small (i.e., k is relatively large), starting the algorithm with an approximate optimal solution for the D-optimal design

problem (for the same data set) can decrease the number of iterations. Since obtaining such a solution is relatively cheap (see Chapter 2), this may decrease the computation time significantly.

Lemma 4.3.4 *Let u^D be a δ-primal feasible solution for the D-optimal design (as defined in Chapter 2), then u^D is an $(\frac{n-k}{k} + \frac{n}{k}\delta)$-primal feasible solution for the D_k-optimal design.*

Proof: For all $1 \leq j \leq m$, we have

$$\omega_j(u) \leq \xi_j(u) \leq (1+\delta)n = \left(1 + \left(\frac{n-k}{k} + \frac{n}{k}\delta\right)\right)k.$$

This proves that u^D is an $(\frac{n-k}{k} + \frac{n}{k}\delta)$-primal feasible solution for $(\mathcal{D})$. □

We will refer to the algorithm which replaces u^0 in Step 1 of Algorithm 4.2.1 with a 1-approximate solution for the D-optimal design problem obtained using WA-TY algorithm (from Chapter 2) as Algorithm 4.2.1-MV.

Lemma 4.3.5 *Under the assumptions of Lemma 4.3.2. Algorithm 4.2.1-MV obtains an ϵ-approximate optimal solution in at most*

$$L(\epsilon) = O(n \ln n + k(\epsilon^{-1} + \ln k + \ln \ln n/k)) \tag{4.3.56}$$

iterations.

Proof: With the new initial solution (4.3.51) becomes $\delta_0 \leq k \ln \frac{2n}{k}$; furthermore the initial solution has at most $O(n \ln n)$ positive components (see [18]). Keeping these changes in mind, the rest of the proof is the same as that of Lemma 4.3.3. □

Note that this bound is independent of the number of points in the data set and can be exploited when dealing with large data sets.

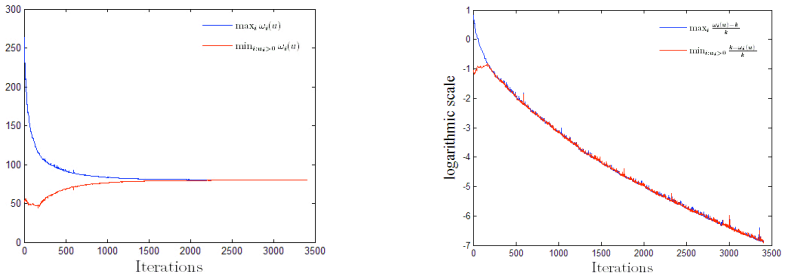

Figure 4.1: Behavior of the algorithm for $(m, n, k) = (10000, 100, 80)$.

4.4 Local Convergence of the Algorithm

In this section, we will show that Algorithm 4.2.1 is locally linearly convergent, i.e., the number of iterations grows with $O(\ln \epsilon^{-1})$ not $O(\epsilon^{-1})$ asymptotically under certain assumptions. The typical behavior of the algorithm as shown in Figure 4.1 illustrates this property. Unfortunately, this bound depends on the data of the problem as well as the dimensions and the constant C in Lemma 4.3.2, and so does not provide a global complexity bound better than that above.

We use the following perturbation of $(\mathcal{P}')$:

$$\begin{array}{rrcl} & \min_{H'>0} \ \bar{f}(H', E) & := & -\ln\det H' \\ (\mathcal{P}'(\kappa)) & (y_i + Ez_i)^T H'(y_i + Ez_i) & \leq & k + \kappa_i, \quad i = 1, \ldots, m. \end{array}$$

Given a dual feasible u which with its associated $K = K(u)$ and $E = E(u)$ satisfies the γ-approximate optimality conditions, let $H' = K^{-1}$ and define $\kappa := \kappa(u, \gamma) \in \mathbb{R}^m$ by

$$\kappa_i := \begin{cases} \gamma k & \text{if } u_i = 0 \\ (y_i + Ez_i)^T K^{-1}(y_i + Ez_i) - k & \text{else.} \end{cases}$$

Observe that each component of κ has absolute value at most γk, and that this property may fail if we merely assume that (u, K, E) is γ-primal feasible. More-

over using (4.3.1),

$$u^T \kappa = \sum_{i:u_i>0} u_i \kappa_i = u^T \omega(u) - k e^T u = k - k = 0. \tag{4.4.57}$$

Lemma 4.4.1 *Suppose* (u, K, E) *satisfies the* γ*-approximate optimality conditions. Then* (H', E) *is optimal in* $(\mathcal{P}'(\kappa(u, \gamma)))$.

Proof: We note that (H', E) is feasible. Furthermore, u provides the required vector of Lagrange multipliers that satisfy the necessary and sufficient conditions for optimality given in the proof of Theorem 4.1.1 for $(\mathcal{P}'(\kappa(u, \gamma)))$. □

As discussed in Section 4.1 and proved in Lemma 4.1.1, there is an equivalent convex problem, say $(\mathcal{P}(\kappa))$, to $(\mathcal{P}'(\kappa))$. Let $\phi(\kappa)$ denote the value function, the optimal value of $(\mathcal{P}(\kappa))$. Then ϕ is convex, and if u' is any vector of Lagrange multipliers for the optimal solution of $(\mathcal{P}(\kappa))$, then u' is a subgradient of ϕ at κ. In particular, if u_* is any vector of Lagrange multipliers for the optimal solution of $(\mathcal{P})$, then u_* is a subgradient of ϕ at 0.

For any u satisfying the γ-approximate optimality conditions and $\kappa := \kappa(u, \gamma)$,

$$\begin{aligned} g(u) = \bar{f}(K^{-1}, E) = \phi(\kappa) &\geq \phi(0) + u_*^T \kappa \\ &= g^* + (u_* - u)^T \kappa \\ &\geq g^* - \|u - u_*\| \|\kappa\|. \end{aligned} \tag{4.4.58}$$

Here the last equality follows from (4.4.57). We have already noted that $\|\kappa\| \leq k\sqrt{m}\gamma$. To obtain an improvement on Lemma 4.3.2, we would like to prove that

$$\|u - u_*\| \leq L\|\kappa\| \leq Lk\sqrt{m}\gamma \tag{4.4.59}$$

whenever $\|\kappa\|$ is sufficiently small. We will use the following assumption:

Assumption 4.4.1 *The strong second-order sufficient condition for local optimality and the linear independence of the active constraints hold for the optimal solution* (H'^*, E^*) *for problem* $(\mathcal{P}')$ *and the corresponding multipliers* u_*.

We have shown that the optimal solutions for problem $(\mathcal{P}'(\kappa))$ and points that satisfy the corresponding KKT system coincide (see the proof of Theorem 4.1.1). If Assumption 4.4.1 holds, we know that u_*, the vector of multipliers for the optimal solution, is unique. Furthermore, the set-valued map S_{KKT}, which maps a perturbation vector κ to the set of optimal solutions for problem $(\mathcal{P}'(\kappa))$ (and corresponding solutions for $(\mathcal{P}(\kappa))$) and their corresponding multipliers, is locally single-valued and locally upper Lipschitz around $(0, (H'^*, E^*), u_*)$ (see Robinson [24] or Theorem 4.2 of Dontchev and Rocakfellar [6]). Also, the set-valued map X_{KKT}, which maps a perturbation vector κ to the set of optimal solutions is single-valued around $(0, (H'^*, E^*))$ (see Corollary 3.5 in Dontchev and Rocakfellar [6]). There exist neighborhoods V around 0 and $W_1 \times W_2$ around $((H'^*, E^*), u_*)$, such that the S_{KKT} map is single-valued in $W_1 \times W_2$ for all $\kappa \in V$ and the X_{KKT} map is single-valued in W_1 for all $\kappa \in V$. Therefore, when γ is small enough so that κ lies in V, there must exist a pair of optimal solutions $(\hat{H}', \hat{E})$ for $(\mathcal{P}'(\kappa))$ (and also $\hat{H}$ for $(\mathcal{P}(\kappa))$) and multipliers $\hat{u}$ such that $(\hat{H}', \hat{E})$ is the only solution in W_1 for $(\mathcal{P}'(\kappa))$ and $((\hat{H}', \hat{E}), \hat{u})$ is the only solution-multiplier pair in $W_1 \times W_2$. If $\hat{u} = u$, the local Lipschitz property of the S_{KKT} map provides that $\|u - u_*\| \leq L\|\kappa\| \leq Lk\sqrt{m}\gamma$ and we are done. Now assume that $\hat{u} \neq u$. There are two cases, $(H', E) = (\hat{H}', \hat{E})$ and $(H', E) \neq (\hat{H}', \hat{E})$. We will show that both cases lead to a contradiction and hence $\hat{u}$ must be equal to u. When $(H', E) \neq (\hat{H}', \hat{E})$, we can come up with two different solutions H and $\hat{H}$ for the convex problem $(\mathcal{P}'(\kappa))$ which are both optimal. Then any convex combination of the points H and $\hat{H}$ must be optimal, too. We can find an optimal solution $\tilde{H}$ arbitrarily close

to $\hat{H}$; hence we can find an optimal solution $(\tilde{H}', \tilde{E})$ for $(\mathcal{P}'(\kappa))$, which is arbitrarily close to $(\hat{H}', \hat{E})$. This violates the local single-valuedness property of the X_{KKT} map at $(\hat{H}', \hat{E})$, so we must have $(H', E) = (\hat{H}', \hat{E})$. On the other hand, if we have two different vectors of multipliers, u and $\hat{u}$, both corresponding to the primal optimal solution (H', E), then we can find another vector of multipliers, say $\tilde{u}$, arbitrarily close to $\hat{u}$. This violates the local uniqueness property of the S_{KKT} map and hence leads to a contradiction. Hence, we have shown that $\hat{u} = u$ whenever Assumption 4.4.1 holds and (4.4.59) is valid.

From (4.4.59) and (4.4.58), we conclude

Proposition 4.4.1 *If Assumption 4.4.1 holds, there is some constant $M > 0$ (depending on the data of problem $(\mathcal{P})$) such that, whenever (u, K, E) is a γ-approximate optimal solution for some sufficiently small γ, we have*

$$g^* - g(u) \leq M\gamma^2. \tag{4.4.60}$$

Let h and $\mathcal{H}$ be defined as in the proof of Lemma 4.3.3. Applying Proposition 4.4.1 instead of Lemma 4.3.2 in (4.3.53), we obtain

$$h(\gamma) \leq \frac{M\gamma^2}{\hat{c}\gamma^2/8} = \frac{8M}{\hat{c}} \text{ for sufficiently small } \gamma, \tag{4.4.61}$$

and this yields, using the argument above (4.3.54), the existence of a constant $Q > 0$ with

$$\mathcal{H}(\epsilon) \leq Q + \frac{16M}{\hat{c}} \ln(1/\epsilon) \text{ for sufficiently small } \epsilon.$$

We therefore have

Theorem 4.4.1 *If Assumption 4.4.1 holds, then there exist data-dependent constants $\tilde{Q}$ and $\tilde{M}$ such that Algorithm 4.2.1 requires at most $\tilde{Q} + \tilde{M} \ln(1/\epsilon)$ iterations to obtain an ϵ-approximate optimal solution.*

4.5 Rank-Deficient Case

As we have briefly discussed above, providing a complete algorithm for the MAEC problem is problematic. Assume we are at the l^{th} iteration with $ZU^{l-1}Z^T$ nonsingular, and produce a new iterate u^l with ZU^lZ^T singular (we will show below that this can only occur at a drop iteration). It is then far from clear how we can continue the algorithm, because our update formulae in Section 4.2 assume nonsingularity of ZU^hZ^T for $h = 1, 2, \ldots$. In this section we show how the algorithm can be modified to deal with this case. We will use $r = n - k$ in the rest of this chapter.

Lemma 4.5.1 *Given* $Z = [z_1, \ldots, z_m] \in \mathbb{R}^{r\times m}$, $\text{range}\,(ZUZ^T) = \text{span}\,(\{z_i : u_i > 0\})$. *Hence* $\text{rank}\,(ZUZ^T) = \dim \text{span}\,(\{z_i : u_i > 0\})$.

Proof: It is a direct consequence of the singular value decomposition that for any matrix $\tilde{Z}$ we have $\text{range}\,(\tilde{Z}) = \text{range}\,(\tilde{Z}\tilde{Z}^T)$. Let Z^+ be a matrix whose columns are the elements of the set $\{z_i : u_i > 0\}$ and U^+ ($U^{+,1/2}$) be a diagonal matrix with (the square roots of) the positive u_i's on the diagonal. Substituting $\tilde{Z} = Z^+U^{+,1/2}$ gives the desired result since

$$\begin{aligned} \text{range}\,(ZUZ^T) &= \text{range}\,(Z^+U^+Z^{+T}) = \text{range}\,(Z^+U^{+,1/2}) \\ &= \text{range}\,(Z^+) = \text{span}\,(\{z_i : u_i > 0\}). \end{aligned} \tag{4.5.62}$$

□

The result above shows that difficulties only occur at drop iterations. To simplify the discussion, we rename u^{l-1} as $\hat{u}$ and let $u = \frac{1}{1-\hat{u}_d}(\hat{u} - \hat{u}_d e_d)$, so that x_d is dropped. We suppose ZUZ^T is singular, so that there are many solutions to $YUZ^T = -EZUZ^T$. However, one is particularly easy to find:

Lemma 4.5.2 *Let $\hat{E}$ satisfy $Y\hat{U}Z^T + \hat{E}Z\hat{U}Z^T = 0$. If $\operatorname{rank}(Z\hat{U}Z^T) = r$ and $\operatorname{rank}(ZUZ^T) = r-1$, then we have the following equalities:*

i. $y_d + \hat{E}z_d = 0$;

ii. $YUZ^T + \hat{E}ZUZ^T = 0$;

iii. $K(u) = \frac{1}{1-\hat{u}_d}K(\hat{u})$; *and*

iv. $\hat{u}_d\zeta_d(\hat{u}) = 1$.

Proof: First observe that $\hat{u} = (1-\hat{u}_d)u + \hat{u}_d e_d$. Since ZUZ^T is singular and $XUX^T \geq 0$, there exists a vector $w \neq 0 \in \mathbb{R}^r$ such that $ZUZ^Tw = 0$ and $YUZ^Tw = 0$. Then we have

$$\begin{aligned}
\hat{u}_d(y_d + \hat{E}z_d)z_d^Tw &= \hat{u}_d y_d z_d^T w + \hat{u}_d \hat{E} z_d z_d^T w \\
&= (1-\hat{u}_d)YUZ^Tw + \hat{u}_d y_d z_d^T w + \dots \\
&\qquad (1-\hat{u}_d)\hat{E}ZUZ^Tw + \hat{u}_d\hat{E}z_d z_d^T w \\
&= Y\hat{U}Z^Tw + \hat{E}Z\hat{U}Z^Tw = (Y\hat{U}Z^T + \hat{E}Z\hat{U}Z^T)w = 0.
\end{aligned}$$

Since $\hat{u}_d > 0$, we must have either $y_d + \hat{E}z_d = 0$ or $z_d^Tw = 0$. Assume that $z_d^Tw = 0$: then we have $Z\hat{U}Z^Tw = ((1-\hat{u}_d)ZUZ^T + \hat{u}_d z_d z_d^T)w = 0$, which contradicts the nonsingularity of $Z\hat{U}Z^T$. Hence we must have $y_d + \hat{E}z_d = 0$ as claimed in (*i*). $Y\hat{U}Z^T + \hat{E}Z\hat{U}Z^T = 0$ and (*i*) imply that

$$\begin{aligned}
YUZ^T + \hat{E}ZUZ^T &= \frac{1}{1-\hat{u}_d}(Y\hat{U}Z^T + \hat{E}Z\hat{U}Z^T - \hat{u}_d y_d z_d^T - \hat{u}_d\hat{E}z_d z_d^T) \\
&= \frac{1}{1-\hat{u}_d}(Y\hat{U}Z^T + \hat{E}Z\hat{U}Z^T - \hat{u}_d(y_d + \hat{E}z_d)z_d^T) = 0,
\end{aligned}$$

from which we get (*ii*). Similarly, we have

$$\begin{aligned}
K(u) &= \frac{1}{1-\hat{u}_d}(Y\hat{U}Y^T + \hat{E}Z\hat{U}Y^T - u_d(y_d + \hat{E}z_d)y_d^T) \\
&= \frac{1}{1-\hat{u}_d}(Y\hat{U}Y^T + \hat{E}Z\hat{U}Y^T) = \frac{1}{1-\hat{u}_d}K(\hat{u}),
\end{aligned}$$

which proves (*iii*). In order to prove (*iv*), we know that $\text{rank}\,(ZUZ^T) = r - 1$ implies that $\det(ZUZ^T) = 0$. We have

$$\begin{aligned}\det(ZUZ^T) &= (1-\hat{u}_d)^{-r}\det(Z\hat{U}Z^T - \hat{u}_d z_d z_d^T) \\ &= (1-\hat{u}_d)^{-r}\det(Z\hat{U}Z^T)(1-\hat{u}_d\zeta_d(\hat{u})) = 0.\end{aligned}$$

Since $\det(Z\hat{U}Z^T) > 0$, we must have (*iv*). Note that the reverse claim is also correct, i.e., $\hat{u}_d\zeta_d(\hat{u}) = 1$ implies that ZUZ^T is singular. □

This result can be generalized as follows:

Lemma 4.5.3 *Let $\hat{u}$ be a dual feasible solution that satisfies $\text{rank}\,(Z\hat{U}Z^T) = r$ with associated matrix $\hat{E}$ such that $Y\hat{U}Z^T + \hat{E}Z\hat{U}Z^T = 0$. Define $R(\hat{u}) := \{i : \text{rank}\,(Z\hat{U}Z^T - \hat{u}_i z_i z_i^T) < r\}$ and suppose $R \subseteq R(\hat{u})$ is such that $\sigma := 1 - \sum_{i\in R}\hat{u}_i > 0$. Let $u = \frac{1}{\sigma}(\hat{u} - \sum_{i\in R}\hat{u}_i e_i)$; then*

i. $YUZ^T + \hat{E}ZUZ^T = 0$ *and*

ii. $K(u) = \frac{1}{\sigma}K(\hat{u})$.

Proof: The proof is very similar to that of the previous lemma. We have $y_i + \hat{E}z_i = 0$ for all $i \in R$ from Lemma 4.5.2, so that

$$\begin{aligned}YUZ^T + \hat{E}ZUZ^T &= \frac{1}{\sigma}(Y\hat{U}Z^T + \hat{E}Z\hat{U}Z^T - \sum_{i\in R}\hat{u}_i(y_i + \hat{E}z_i)z_i^T) \\ &= \frac{1}{\sigma}(Y\hat{U}Z^T + \hat{E}Z\hat{U}Z^T) = 0,\end{aligned}$$

which gives (*i*). Similarly,

$$\begin{aligned}K(u) &= \frac{1}{\sigma}(Y\hat{U}Y^T + \hat{E}Z\hat{U}Y^T - \sum_{i\in R}u_i(y_i + \hat{E}z_i)y_i^T) \\ &= \frac{1}{\sigma}(Y\hat{U}Y^T + \hat{E}Z\hat{U}Y^T) = \frac{1}{\sigma}K(\hat{u}),\end{aligned}$$

and hence we get (*ii*). □

Let u be the current iterate, and assume that we have $\hat{u}$ such that $Z\hat{U}Z^T$ is nonsingular and $u = \frac{1}{\sigma}(\hat{u} - \sum_{i\in R}\hat{u}_i e_i)$ for some index set $R \subseteq R(\hat{u})$ and $\sigma = 1 - \sum_{i\in R}\hat{u}_i$. Also suppose E satisfies $Y\hat{U}Z^T + EZ\hat{U}Z^T = 0$. If $R \neq R(\hat{u})$, it is immediate from Lemma 4.5.3 that replacing u by $\frac{1}{\sigma}(\hat{u} - \sum_{i\in R(\hat{u})}\hat{u}_i e_i)$ increases $\ln\det(K(u))$, so we can assume that $R = R(\hat{u})$. The lemma above shows that much of the information required related to the current iterate u follows from information for the associated vector $\hat{u}$ for which $Z\hat{U}Z^T$ is nonsingular. We refer to iterations where we drop points x_d for $d \in R$ as *deferred* updates since we maintain information such as ω, ζ, K, etc., for the vector $\hat{u}$, which is the dual vector before dropping the points x_d, $d \in R$.

Now, assume that we make an update of the form $u_+ = (1-\tau)u + \tau e_j$. If we set $\hat{\tau} = \frac{\tau\sigma}{1-\tau+\tau\sigma}$, then the update $\hat{u}_+ = (1-\hat{\tau})\hat{u} + \hat{\tau}e_j$ will satisfy $u_+ = \frac{1}{\sigma_+}(\hat{u}_+ - \sum_{i\in R}\hat{u}_{+i}e_i)$ where $\sigma_+ = 1 - \sum_{i\in R}\hat{u}_{i+} = \frac{\sigma}{1-\tau+\tau\sigma}$. We can assume that $j \notin R$ without loss of generality because $j \in R$ implies that $y_j + Ez_j = 0$ which in turn implies $\omega_j(\hat{u}) = 0$, and such a choice of j would never be made by our algorithm. Let us assume for now that $R(\hat{u}_+) = R$.

Letting $\hat{\lambda} = \frac{\hat{\tau}}{1-\hat{\tau}}$ (which implies that $\hat{\lambda} = \sigma\lambda$), and using the update formulae in Section 4.2, we obtain

$$K(u_+) \;=\; \frac{(1-\hat{\tau})}{\sigma(1-\hat{\tau})+\hat{\tau}}\left(K(\hat{u}) + \frac{\hat{\lambda}}{1+\hat{\lambda}\zeta_j(\hat{u})}(y_j+Ez_j)(y_j+Ez_j)^T\right).$$

We have

$$\begin{aligned}\ln\det K(u_+) &= \ln\det K(\hat{u}) - k\ln\left(\frac{\sigma(1-\hat{\tau})+\hat{\tau}}{1-\hat{\tau}}\right) + \ln\left(1+\frac{\hat{\lambda}\omega_j(\hat{u})}{1+\hat{\lambda}\zeta_j(\hat{u})}\right)\\ &= \ln\det K(\hat{u}) - k\ln\left(\sigma+\hat{\lambda}\right) + \ln\left(1+\frac{\hat{\lambda}\omega_j(\hat{u})}{1+\hat{\lambda}\zeta_j(\hat{u})}\right),\end{aligned}$$

whose derivative is

$$\begin{aligned}\frac{\partial \ln\det K(u_+)}{\partial\hat{\lambda}} &= -\frac{k}{\sigma+\hat{\lambda}} + \frac{1}{1+\frac{\hat{\lambda}\omega_j(\hat{u})}{1+\hat{\lambda}\zeta_j(\hat{u})}}\left(\frac{\omega_j(\hat{u})}{1+\hat{\lambda}\zeta_j(\hat{u})} - \frac{\hat{\lambda}\zeta_j(\hat{u})\omega_j(\hat{u})}{(1+\hat{\lambda}\zeta_j(\hat{u}))^2}\right)\\ &= -\frac{k}{\sigma+\hat{\lambda}} + \frac{\omega_j(\hat{u})}{(1+\hat{\lambda}(\zeta_j(\hat{u})+\omega_j(u)))(1+\hat{\lambda}\zeta_j(\hat{u}))}\\ &= \frac{-k}{(\sigma+\hat{\lambda})(1+\hat{\lambda}\zeta_j(\hat{u}))(1+\hat{\lambda}(\zeta_j(\hat{u})+\omega_j(\hat{u})))}\left(\hat{a}\hat{\lambda}^2 - 2\hat{b}\hat{\lambda}+\hat{c}\right),\end{aligned} \tag{4.5.63}$$

where $\hat{a} := \zeta_j(\hat{u})(\zeta(\hat{u})+\omega(\hat{u})) \geq 0$, $\hat{b} := -\zeta_j(\hat{u}) - \frac{\omega_j(\hat{u})}{2} + \frac{\omega_j(\hat{u})}{2k} \leq 0$, and $\hat{c} := 1 - \frac{\sigma\omega_j(\hat{u})}{k}$. Note that this derivative at $\hat{\lambda} = 0$ is positive or negative according as $\hat{c}$ is negative or positive, or equivalently according as $\sigma\omega_j(\hat{u})$ (which is $\omega_j(u) = (y_j + Ez_j)^T K(u)^{-1}(y_j + Ez_j)$ by Lemma 4.5.3) is greater or less than k, as in the full-rank case.

Now we can try to find the optimal step length that maximizes the objective function value of the new iterate u_+ as before. We need to be careful at this point. If the choice of the vertex x_j used in the update and the corresponding optimal step length results in an increase, decrease, or drop iteration, then we must have $R(\hat{u}_+) = R(\hat{u}) = R$, and hence performing this update is the best option we have available. (Note that a drop iteration cannot lose rank, because we have already dropped all such points by our assumption that $R = R(\hat{u})$.) On the other hand, if the current update turns out to be an add iteration, we may have $R(\hat{u}_+) \neq R$ and we can perform one of the deferred updates in the list R at the same time as we perform the add iteration. As we discuss below, an appropriate choice of step lengths then results in a new iterate u_+ identical to the current iterate u, but we will find a new $\hat{u}_+$ and a new matrix E_+ which satisfies $YU_+Z^T + E_+ZU_+Z^T = 0$. This feature allows the continued progress of the algorithm.

Assume we would like to perform an add iteration in which $\hat{u}_j$ will be increased from 0 to a positive value. This may result in a new iterate for which at least one of the deferred updates, say that corresponding to dropping x_d, may be performed without violating the nonsingularity assumption. If this is the case, we will consider a combined update which simultaneously adds x_j and drops x_d, with d in the deferred update list R. Therefore updates of the following form will be of interest:

$$\begin{aligned}\hat{u}_+ &= (1-\hat{\tau})\left(\frac{\hat{u}-\hat{u}_d e_d}{1-\hat{u}_d}\right)+\hat{\tau}e_j \\ &= \left(\frac{1-\hat{\tau}}{1-\hat{u}_d}\right)\hat{u}-\frac{(1-\hat{\tau})\hat{u}_d}{1-\hat{u}_d}e_d+\hat{\tau}e_j.\end{aligned}$$

Let $\hat{\lambda}=\frac{\hat{\tau}(1-\hat{u}_d)}{1-\hat{\tau}}$, so that we can write the update above as

$$\hat{u}_+ = (1-\hat{u}_d+\hat{\lambda})^{-1}(\hat{u}-\hat{u}_d e_d+\hat{\lambda}e_j). \tag{4.5.64}$$

Let us see when such a deferred drop can be made. This will be the case exactly when the corresponding matrix $Z\hat{U}Z^T+\hat{\lambda}z_j z_j^T-\hat{u}_d z_d z_d^T$ is nonsingular (since the factor $1-\hat{u}_d+\hat{\lambda}$ is always positive). Now we use

$$(Z\hat{U}Z^T+\hat{\lambda}z_j z_j^T)^{-1}=(Z\hat{U}Z^T)^{-1}-\hat{\lambda}(1+\hat{\lambda}\zeta_j(\hat{u}))^{-1}(Z\hat{U}Z^T)^{-1}z_j z_j^T(Z\hat{U}Z^T)^{-1}$$

to see that the above matrix is nonsingular iff $1\neq\hat{u}_d z_d^T(Z\hat{U}Z^T+\hat{\lambda}z_j z_j^T)^{-1}z_d=\hat{u}_d\zeta_d(\hat{u})-\hat{\lambda}\hat{u}_d\zeta_{dj}(\hat{u})^2(1+\hat{\lambda}\zeta_j(\hat{u}))^{-1}$. Thus, using $\hat{u}_d\zeta_d(\hat{u})=1$, we see that x_d can be dropped exactly when $\zeta_{dj}(\hat{u})$ is nonzero. If all $\zeta_{dj}(\hat{u})$, $d\in R$, are zero, we proceed with the add iteration without performing any deferred updates. Suppose now that one of these quantities is nonzero. Then we choose a corresponding d and update $\hat{u}$ as in (4.5.64). Note that Lemmas 4.5.1 and 4.5.3 show that $Z\hat{U}Z^T-\hat{u}_d z_d z_d^T-\hat{u}_{d'}z_{d'}z_{d'}^T$ has rank $r-2$ for any two indices d and d' in R, so we can only perform at most one deferred update.

If $\hat{u}$ is updated as in (4.5.64), then we have

$$Y\hat{U}_+Y^T = (1-\hat{u}_d+\hat{\lambda})^{-1}(Y\hat{U}Y^T - \hat{u}_d y_d y_d^T + \hat{\lambda} y_j y_j^T), \tag{4.5.65}$$

$$Y\hat{U}_+Z^T = (1-\hat{u}_d+\hat{\lambda})^{-1}(Y\hat{U}Z^T - \hat{u}_d y_d z_d^T + \hat{\lambda} y_j z_j^T), \tag{4.5.66}$$

and

$$Z\hat{U}_+Z^T = (1-\hat{u}_d+\hat{\lambda})^{-1}(Z\hat{U}Z^T - \hat{u}_d z_d z_d^T + \hat{\lambda} z_j z_j^T). \tag{4.5.67}$$

We would like to find a matrix E_+ that satisfies $E_+Z\hat{U}_+Z^T + Y\hat{U}_+Z^T = 0$. Let us assume that it can be chosen of the form $E_+ = E + pq^T$ for some $p \in \mathbb{R}^k$ and $q \in \mathbb{R}^r$. Keeping in mind that $EZ\hat{U}Z^T = -Y\hat{U}Z^T$ and $y_d + Ez_d = 0$ (see Lemmas 4.5.3 and 4.5.2), we can find appropriate vectors p and q as follows. We would like to satisfy:

$$\begin{aligned}
E_+Z\hat{U}_+Z^T &= -Y\hat{U}_+Z^T, \text{ or} \\
(E+pq^T)(Z\hat{U}Z^T - \hat{u}_d z_d z_d^T + \hat{\lambda} z_j z_j^T) &= -(Y\hat{U}Z^T - \hat{u}_d y_d z_d^T + \hat{\lambda} y_j z_j^T), \text{ or} \\
\hat{\lambda}(y_j + Ez_j)z_j^T &= -p(q^T(Z\hat{U}Z^T - \hat{u}_d z_d z_d^T + \hat{\lambda} z_j z_j^T)).
\end{aligned}$$

Setting $p = y_j + Ez_j$, and $q = \pi(Z\hat{U}Z^T)^{-1}z_d$, this holds as long as

$$-\hat{\lambda} z_j = \pi z_d - \hat{u}_d \pi \zeta_d(\hat{u}) z_d + \hat{\lambda}\pi\zeta_{dj}(\hat{u}) z_j = \hat{\lambda}\pi\zeta_{dj}(\hat{u}) z_j$$

where we have used $\hat{u}_d\zeta_d(\hat{u}) = 1$ since $d \in R$. This requires that $\pi = -\frac{1}{\zeta_{dj}(\hat{u})}$, and we have therefore found a matrix

$$\begin{aligned}
E_+ &= E + pq^T \\
&= E - \frac{1}{\zeta_{dj}(\hat{u})}(y_j + Ez_j)z_d^T(Z\hat{U}Z^T)^{-1},
\end{aligned}$$

which can be used in further arguments. Using again the fact that $\hat{u}_d\zeta_d(\hat{u}) = 1$ and also that $y_d + Ez_d = 0$, we get

$$K(\hat{u}_+) = Y\hat{U}_+Y^T + E_+Z\hat{U}Y^T$$

$$
\begin{aligned}
&= (1-\hat{u}_d+\hat{\lambda})^{-1}(Y\hat{U}Y^T - \hat{u}_d y_d y_d^T + \hat{\lambda} y_j y_j^T + \dots \\
&\qquad (E+pq^T)(Z\hat{U}Y^T - \hat{u}_d z_d y_d^T + \hat{\lambda} z_j y_j^T)) \\
&= (1-\hat{u}_d+\hat{\lambda})^{-1}(K(\hat{u}) + \hat{\lambda}(y_j+Ez_j)y_j^T - \dots \\
&\qquad \frac{1}{\zeta_{dj}(\hat{u})}(y_j+Ez_j)(-z_d^T E - \hat{u}_d\zeta_d(\hat{u})y_d^T + \hat{\lambda}\zeta_{dj}(\hat{u})y_j^T)) \\
&= (1-\hat{u}_d+\hat{\lambda})^{-1}(K(\hat{u}) + \hat{\lambda}(y_j+Ez_j)y_j^T - \dots \\
&\qquad \frac{1}{\zeta_{dj}(\hat{u})}(y_j+Ez_j)((1-\hat{u}_d\zeta_d(\hat{u}))y_d^T + \hat{\lambda}\zeta_{dj}(\hat{u})y_j^T)) \\
&= (1-\hat{u}_d+\hat{\lambda})^{-1}K(\hat{u}). \qquad (4.5.68)
\end{aligned}
$$

Since x_d will be dropped, it appears that we will have $R_+ = R - \{d\}$ which yields $\sigma_+ = 1 - \sum_{i\in R_+}\hat{u}_{i+} = \frac{1-\hat{\tau}}{1-\hat{u}_d}\sigma + \hat{\tau} = \frac{\sigma+\hat{\lambda}}{1-\hat{u}_d+\hat{\lambda}}$. Therefore we can write the new objective function value as

$$
\begin{aligned}
\ln\det K(u_+) &= \ln\det\frac{1}{\sigma+\hat{\lambda}}K(\hat{u}) \\
&= \ln\det K(\hat{u}) - k\ln(\sigma+\hat{\lambda}), \qquad (4.5.69)
\end{aligned}
$$

which is maximized at $\hat{\lambda} = \hat{\tau} = 0$. However, if we choose $\hat{\tau}$ to be zero, we find that $Z\hat{U}_+Z^T$ is singular. We deal with this situation by setting $\hat{\tau}$ positive, but then as we will see letting x_j be a deferred drop instead of x_d.

As before, we can update $\omega(\hat{u}_+)$ cheaply:

$$
\begin{aligned}
\omega_i(\hat{u}_+) &= (y_i+E_+z_i)^T K(\hat{u}_+)^{-1}(y_i+E_+z_i) \\
&= (1-\hat{u}_d+\hat{\lambda})\left(y_i + (E - \frac{1}{\zeta_{dj}(\hat{u})}(y_j+Ez_j)z_d^T(ZUZ^T)^{-1})z_i\right)^T K(\hat{u})^{-1} \\
&\qquad \dots\left(y_i + (E - \frac{1}{\zeta_{dj}(\hat{u})}(y_j+Ez_j)z_d^T(ZUZ^T)^{-1})z_i\right) \\
&= (1-\hat{u}_d+\hat{\lambda})\left(y_i + Ez_i - \frac{\zeta_{di}(\hat{u})}{\zeta_{dj}(\hat{u})}(y_j+Ez_j)\right)^T K(\hat{u})^{-1} \\
&\qquad \dots\left(y_i + Ez_i - \frac{\zeta_{di}(\hat{u})}{\zeta_{dj}(\hat{u})}(y_j+Ez_j)\right)
\end{aligned}
$$

$$= (1 - \hat{u}_d + \hat{\lambda})\left(\omega_i(\hat{u}) - \frac{2\zeta_{di}(\hat{u})}{\zeta_{dj}(\hat{u})}\omega_{ij}(\hat{u}) + \frac{\zeta_{di}^2(\hat{u})}{\zeta_{dj}^2(\hat{u})}\omega_j(\hat{u})\right). \quad (4.5.70)$$

Furthermore we have

$$\begin{aligned}(Z\hat{U}_+Z^T)^{-1} &= (1 - \hat{u}_d + \hat{\lambda})(Z\hat{U}Z^T - \hat{u}_d z_d z_d^T + \hat{\lambda} z_j z_j^T)^{-1} \\ &= (1 - \hat{u}_d + \hat{\lambda})((Z\hat{U}Z^T)^{-1} + \frac{1 + \hat{\lambda}\zeta_j(\hat{u})}{\hat{\lambda}\zeta_{dj}^2(\hat{u})}(Z\hat{U}Z^T)^{-1} z_d z_d^T (Z\hat{U}Z^T)^{-1} \\ &\quad \ldots - \frac{1}{\zeta_{dj}(\hat{u})}(Z\hat{U}Z^T)^{-1}(z_d z_j^T + z_j z_d^T)(Z\hat{U}Z^T)^{-1}), \quad (4.5.71)\end{aligned}$$

which leads to

$$\zeta_i(\hat{u}_+) = (1 - \hat{u}_d + \hat{\lambda})\left(\zeta_i(\hat{u}) + \frac{(1 + \hat{\lambda}\zeta_j(\hat{u}))\zeta_{di}^2(\hat{u})}{\hat{\lambda}\zeta_{dj}^2(\hat{u})} - \frac{2\zeta_{di}(\hat{u})\zeta_{ij}(\hat{u})}{\zeta_{dj}(\hat{u})}\right). \quad (4.5.72)$$

We have $\hat{u}_{+j}\zeta_j(\hat{u}_+) = \hat{\tau}\frac{(1-\hat{u}_d+\hat{\lambda})}{\hat{\lambda}} = 1$, which implies that $j \in R(\hat{u}_+)$. Therefore we can perform a combined step in which x_j is added while x_d is dropped and add j to the deferred update list. The new deferred update list $R_+ = R - \{d\} \cup \{j\}$ satisfies $R_+ = R(\hat{u}_+)$ since we cannot have any more new deferred updates. The choice of $\hat{\lambda}$ is irrelevant because the new feasible point u_+ has as its components not in R_+ scalings of those of $\hat{u} - \hat{u}_d e_d + \hat{\lambda} e_j$ and so is independent of $\hat{\lambda}$ and in fact equal to u. We prefer to use $\hat{\lambda} = \hat{u}_d$ since it simplifies the calculations (see (4.5.64)) and σ is unchanged.

Based on this discussion we can modify Algorithm 4.2.1 as follows:

Algorithm 4.5.1

Input: $X \in \mathbb{R}^{n\times m}$, $k \in \{1,\ldots,n\}$ *and* $\epsilon > 0$.

Step 0. *Initialize* $u = e/m$, $\hat{u} = u$, $R = \emptyset$ *and* $\sigma = 1$,

compute $K(\hat{u})$, $E(\hat{u})$, $\omega(\hat{u})$, *and* $\zeta(\hat{u})$,

and set $\omega(u) = \omega(\hat{u})$, *and* $\zeta(u) = \zeta(\hat{u})$.

Step 1. *Find* $s := \arg\max_i\{\omega_i(u) - k\}$, $t := \arg\min_i\{\omega_i(u) - k : u_i > 0\}$.

If $\omega_s(u) - k \le \epsilon k$ *and* $\omega_t(u) - k \ge -\epsilon k$,

STOP: *u satisfies the* ϵ*-approximate optimality conditions.*

Step 2. If $\omega_t(u) = 0$, *let* $D = \{i : \hat{u}_i\zeta_i(\hat{u}) = 1\}$,

and set $R = R \cup D$, $\sigma = 1 - \sum_{i\in R}\hat{u}_i$, $u = \sigma^{-1}\hat{u}$, $u(R) = 0$,

$\omega(u) = \sigma\omega(\hat{u})$, *and* $\zeta(u) = \sigma\zeta(\hat{u})$. ***Go to*** *Step 1.*

Step 3. If $\omega_s(u) - k < k - \omega_t(u)$, *set* $j = t$ *and* ***go to*** *Step 6. Else* $j = s$.

Step 4. If $\hat{u}_j = 0$ *and* $R \neq \emptyset$, *calculate* $\zeta_{d'j}(\hat{u})$ *for* $d' \in R$,

and find $d := \arg\max_{d'\in R}\{|\zeta_{d'j}(\hat{u})|\}$.

If $\zeta_{dj}(\hat{u}) = 0$, ***go to*** *Step 5.*

Otherwise, replace $\hat{u}$ *by* $\hat{u} - \hat{u}_d e_d + \hat{u}_d e_j$,

set $R = R - \{d\} \cup \{j\}$, *update* $E(\hat{u})$, $\omega(\hat{u})$, *and* $\zeta(\hat{u})$,

and set $\omega(u) = \sigma\omega(\hat{u})$ *and* $\zeta(u) = \sigma\zeta(\hat{u})$. ***Go to*** *Step 1.*

Step 5. *Replace* $\hat{u}$ *by* $\hat{u}_+ := (1 - \hat{\tau})\hat{u} + \hat{\tau}e_j$, *where* $\hat{\tau} > 0$ *is chosen as in Section 4.5 to maximize* $g(u)$. ***Go to*** *Step 7.*

Step 6. *Replace* $\hat{u}$ *by* $\hat{u}_+ := (1 - \hat{\tau})\hat{u} + \hat{\tau}e_j$, *where now* $\hat{\tau}$ *is chosen from negative values to maximize* $g(u)$ *subject to* $\hat{u}_+$ *remaining feasible.*

Step 7. *Update* $K(\hat{u})$, $E(\hat{u})$, $\omega(\hat{u})$, *and* $\zeta(\hat{u})$,

set $\sigma = 1 - \sum_{i\in R}\hat{u}_i$, $u = \sigma^{-1}\hat{u}$, $u(R) = 0$,

$\omega(u) = \sigma\omega(\hat{u})$ *and* $\zeta(u) = \sigma\zeta(\hat{u})$. ***Go to*** *Step 1.*

This algorithm also starts with a dual feasible solution. We can use any initial solution u as long as $ZUZ^T \succ 0$. A good candidate is $u = e/m$ since the z_i's span R^r. Each feasible solution vector u is associated with a lifted solution vector $\hat{u}$ satisfying $u = \sigma^{-1}\hat{u}$ and $u_R = 0$. R is the set of deferred updates accumulated since the first iteration and σ is the weight of the remaining components of $\hat{u}$. We work with $\hat{u}$ for which $Z\hat{U}Z^T \succ 0$ by construction, although the solution that we are interested in is u for which ZUZ^T may be singular. As proved in Lemma 4.5.3, $\hat{u}$ provides all the information we need in order to calculate the variance and the objective function associated with u, so that we can figure out the next iterate and step length which maximizes $g(u)$ while taking a Frank-Wolfe step such as $u_+ = (1-\tau)u + e_j$ (or equivalently, $\hat{u}_+ = (1-\hat{\tau})\hat{u} + \hat{\tau}e_j$). At each iteration, we check whether we can add more deferred drop iterations, since each will increase $g(u)$. In some of the iterations, we are able to make a step where we exchange one of the points in the deferred set R with a new point. Although this step does not improve the objective function, it changes the axis of the cylinder at hand which may lead to better iterates. The following example demonstrates how the algorithm works on a toy problem. The cylinders generated at each iteration of the algorithm are illustrated below in Figure 4.2. Note that Algorithm 4.2.1 would fail at the first iteration for this example if we chose $j = 2$.

Example 4.5.1 *For* $X = \begin{bmatrix} 3 & 2 & 0 & 0 & 6 \\ 1 & 2 & 3 & 4 & 0 \end{bmatrix}$ *and* $k = 1$, *Algorithm 4.5.1 obtains a* 10^{-7}*-approximate optimal solution in 3 iterations as shown in Table 4.1.*

We also consider another version of this algorithm in which Step 3 is modified to calculate the possible improvement in the objective function for each vertex (not just the two with the maximal and minimal version as in Algorithm

Table 4.1: Execution of Algorithm 4.5.1 for a 2-D Toy Example

Iteration 0: Initialization

$$\hat{u}_0 = \begin{bmatrix} 0 & 0.5 & 0.5 & 0 & 0 \end{bmatrix} \qquad u_0 = \begin{bmatrix} 0 & 0.5 & 0.5 & 0 & 0 \end{bmatrix}$$

$$\omega(\hat{u}_0) = \begin{bmatrix} 0.889 & 0 & 2 & 3.556 & 8 \end{bmatrix} \qquad \omega(u_0) = \begin{bmatrix} 0.889 & 0 & 2 & 3.556 & 8 \end{bmatrix}$$

$$R_0 = \emptyset \quad E_0 = -1 \quad \sigma_0 = 1 \quad \hat{\tau}_0 = 0 \quad \tau_0 = -1 \quad j_0 = 2$$

Iteration 1: Dropping x_2 is deferred

$$\hat{u}_1 = \begin{bmatrix} 0 & 0.5 & 0.5 & 0 & 0 \end{bmatrix} \qquad u_1 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\omega(\hat{u}_1) = \begin{bmatrix} 0.889 & 0 & 2 & 3.556 & 8 \end{bmatrix} \qquad \omega(u_1) = \begin{bmatrix} 0.445 & 0 & 1 & 1.778 & 4 \end{bmatrix}$$

$$R_1 = \{2\} \quad E_1 = -1 \quad \sigma_1 = 0.5 \quad \hat{\tau}_1 = N/A \quad \tau_1 = N/A \quad j_1 = 5 \quad d_1 = 2$$

Iteration 2: x_2 is dropped and x_5 is deferred

$$\hat{u}_2 = \begin{bmatrix} 0 & 0 & 0.5 & 0 & 0.5 \end{bmatrix} \qquad u_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\omega(\hat{u}_2) = \begin{bmatrix} 0.222 & 0.889 & 2 & 3.556 & 0 \end{bmatrix} \quad \omega(u_2) = \begin{bmatrix} 0.111 & 0.445 & 1 & 1.778 & 0 \end{bmatrix}$$

$$R_2 = \{5\} \quad E_2 = 0 \quad \sigma_2 = 0.5 \quad \hat{\tau}_2 = 1 \quad \tau_2 = 1 \quad j_2 = 4$$

Iteration 3: x_4 is added

$$\hat{u}_3 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad u_3 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\omega(\hat{u}_3) = \begin{bmatrix} 0.063 & 0.25 & 0.563 & 1 & 0 \end{bmatrix} \quad \omega(u_3) = \begin{bmatrix} 0.063 & 0.25 & 0.563 & 1 & 0 \end{bmatrix}$$

$$R_3 = \{5\} \quad E_3 = 0 \quad \sigma_3 = 1 \quad \hat{\tau}_3 = N/A \quad \tau_3 = N/A \quad j_3 = N/A$$

4.5.1. In this version, the vertex which gives the best improvement is chosen. This algorithm will be referred to as Algorithm 4.5.1-ALL. Both versions have attractive features as will be discussed in the next section.

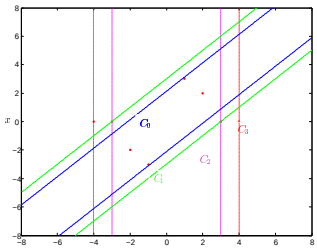

Figure 4.2: Cylinders generated by Algorithm 4.5.1 for Example 4.5.1.

4.6 Computational Study

In order to illustrate the efficiency of the algorithm, we have carried out computational tests with Algorithm 4.5.1 and Algorithm 4.5.1-ALL. The computational experiments were carried out on a 3.40 GHz Pentium IV processor with 1.0 GB RAM using MATLAB version R2006b. The first data set was randomly generated as in [31] with sizes (n, m) varying from $(10, 500)$ to $(100, 100000)$. For each (n, m) value, we have solved the problem for three different values of k ($k = 0.2n, 0.5n, 0.8n$). We have set $\epsilon = 10^{-4}$. For each fixed (n, k, m) ten different problem instances were generated for each data set. The computational results are reported in terms of averages over these instances in Table 4.2, which is divided into three sets of columns. The first set of columns reports the size (n, k, m). The second set of columns presents the results regarding the CPU time and is further divided into two parts, the first of which is devoted to the CPU time in

seconds in order to obtain an $\epsilon-$approximate optimal solution using Algorithm 4.5.1 while the second one displays the CPU time using Algorithm 4.5.1-ALL. The last set of columns reports the number of iterations and also further divided into two columns, displaying the number of iterations needed by Algorithm 4.5.1 and Algorithm 4.5.1-ALL in this order. We have used the initialization algorithm in [18] to find the initial feasible solutions in our experiments. It was observed that these initial solutions decrease the number of iterations needed by the algorithms in practice. Finding a provably better initialization method would improve the theoretical complexity results in Section 4.3 and devising such a method remains as a challenge for us.

Table 4.2: Mean of the numbers of iterations and the solution times of two versions of Algorithm 4.5.1 for random samples of 21 problems, using data sets for Table 2 of Sun and Freund [31].

			CPU Time (Seconds)		No. of Iterations	
n	k	m	Alg 4.5.1	Alg 4.5.1-ALL	Alg 4.5.1	Alg 4.5.1-ALL
10	2	500	0.55	0.81	1047.9	880
10	5	500	0.32	0.56	624.1	623
10	8	500	0.20	0.33	385.2	372.3
10	2	1000	0.55	1.17	1008.6	985.4
10	5	1000	0.56	1.10	979.1	908.4
10	8	1000	0.38	0.86	685	680.4
20	4	5000	4.28	13.20	3300.2	3058.6
20	10	5000	2.51	9.17	2147.1	2189.4
20	16	5000	1.37	5.48	1260.1	1346.7

As demonstrated by Tables 4.2 and 4.3, both algorithms are capable of

Table 4.3: Mean of the numbers of iterations and the solution times of two versions of Algorithm 4.5.1 for random samples of 21 problems, using data sets for Table 2 of Sun and Freund [31].

			CPU Time (Seconds)		No. of Iterations	
n	k	m	Alg 4.5.1	Alg 4.5.1-ALL	Alg 4.5.1	Alg 4.5.1-ALL
20	4	10000	8.37	31.26	3670.7	3917.8
20	10	10000	4.15	16.03	1988.2	2056.3
20	16	10000	3.18	12.67	1577.9	1595.4
30	6	30000	40.49	136.63	5685.7	5771.4
30	15	30000	22.67	78.53	3355.4	3372.2
30	24	30000	17.75	65.86	2812.9	2863.4
50	10	50000	156.02	412.71	9477.5	9159.1
50	25	50000	99.81	287.07	6537.3	6593.7
50	40	50000	68.12	207.37	4983.4	4937.6
100	20	100000	1120.34	2302.59	16826.9	17090.6
100	50	100000	783.38	1700.31	13125.3	13289.5
100	80	100000	555.76	1279.65	10535.4	10540.8

solving very large instances of the problem in a reasonable amount of time. Although using Algorithm 4.5.1-ALL may decrease the number of iterations needed, the total CPU time spent by Algorithm 4.5.1-ALL is always worse than that by Algorithm 4.5.1 due to the large amount of calculation needed at each iteration.

Although the theoretical results suggest that the number of iterations and the total time spent by the algorithms are increasing with k, the experimental results demonstrate that in practice the problem becomes easier as the number

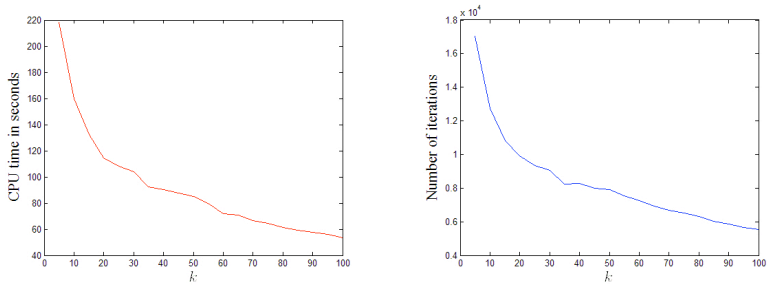

Figure 4.3: Average CPU times (left) and average number of iterations required (right) to obtain a 10^{-4}-approximate optimal solution using Algorithm 4.5.1 for randomly generated data sets with $(n, m) = (100, 10000)$ and various values of k

of parameters to be estimated are increased. This may be explained by the fact that as k increases the MAEC problem becomes increasingly closer in structure to the much easier MVEE problem. It is observed that for a problem instance with fixed $(n, 0.8n, m)$, the number of iterations required to solve the corresponding MVEE problem to the same level of precision is around 90 percent of the number of iterations required to solve the MAEC problem. In order to take a closer look at this phenomenon, we have conducted experiments on 10 data sets, where $(n, m) = (100, 10000)$. The instances are randomly generated as above and k is chosen from the set $\{5, 10, \ldots, 95, 100\}$. The results are summarized in Figure 4.3. In both figures the horizontal axis corresponds to the various values of the parameter k. The vertical axis corresponds to the average CPU time (left) and the average number of iterations (right) spent by the algorithm to solve 10 instances of the problem with parameter set $(100, k, 10000)$ in Figure 4.3. The problem instances are generated as in Table 4.2.

4.7 Discussion and Conclusions

In this chapter, we have developed and analyzed an algorithm for the Minimum-Area Enclosing Ellipsoidal Cylinder problem. We have shown that this problem is a generalization of the Minimum-Volume Enclosing Ellipsoid problem. Our theoretical discussion and computational results show that the more general and harder MAEC problem can be solved by a Frank-Wolfe type algorithm just like the MVEE problem. Unfortunately developing this algorithm and analyzing its properties is not as straightforward as it is for the MVEE problem. We have illustrated this fact using a simple example and suggested a modification of the algorithm. One may suspect that the modified algorithm can cycle for some instances and fail. In our experience this never happens. We are not yet able to prove but we strongly believe that the modified algorithm does not cycle.

Our work shows how to use first-order methods to solve this problem. It is obvious that large instances of the problem can only be attacked by these type of techniques. The addition of Wolfe's away steps makes a huge difference in the number of iterations required to obtain a certain accuracy level in practice. The local convergence result in Section 4.4 provides a theoretical explanation for this phenomenon. Similar behavior is observed for the MVEE problem as discussed in [2].

Note that similar algorithms can be designed for a related cylindrical inclusion problem, which finds an enclosing ellipsoidal cylinder such that the intersection with the subspace Π has the smallest sum of inverses of semi-axes, together with its dual, the A_k-optimal design, which is a generalization of the

A-optimal design problem in Chapter 3. This version of the design problem finds a design vector such that the mean dispersion of the error is minimized when estimating first k parameters of a linear model with n unknown parameters. We have developed such an algorithm that works well for all practical instances so far; nevertheless the details of the analysis and computations are quite cumbersome and omitted for the sake of conciseness.

CHAPTER 5

ANOTHER GENERALIZATION: PARTIAL INCLUSION

Given an integer h, $1 \leq h \leq m$, and an arbitrary data set $\mathcal{X} = \{x_1, \ldots, x_m\} \subset \mathbb{R}^n$, which has m distinct points, the Minimum-Volume Ellipsoid (MVE) estimator is defined to be the minimum-volume ellipsoid that encloses at least h points in the data set. Let $\mathcal{I} = \{1, \ldots, m\}$. The problem of finding the MVE estimator of $\mathcal{X}$ can be formulated as a combinatorial problem, e.g.,

$$\begin{aligned} \min_{H>0, c\in\mathbb{R}^n} \quad & f(H) := -\ln\det H \\ & |\{i \in \mathcal{I} : (x_i - c)^T H (x_i - c) \leq n\}| \geq h, \end{aligned}$$

and also as a MINLP (Mixed Integer Nonlinear Program).

The MVE estimator is an important tool in robust regression and outlier detection in statistics. Rousseeuw and Leroy [26] proved that if $h = \lceil (m + n + 1)/2 \rceil$, then the MVE estimator has the maximum breakpoint (i.e., an outlier detection method based on the MVE estimator can detect outliers in data sets with many contaminated points). The problem of finding the exact MVE estimator is a hard combinatorial problem with two interactive components: finding the optimum subset of at least h points and computing the minimum-volume ellipsoid that encloses this subset.

As discussed in Chapter 2, the problem of computing the minimum-volume enclosing ellipsoid (MVEE) of a given set is well-studied. There are many efficient algorithms in the literature some of which are discussed in depth previously in this thesis.

Unfortunately, the first component, i.e., choosing the best subset, is not an easy task. The number of subsets to be considered is large even for small in-

stances of the problem. For example, when $m = 20$, $n = 2$, and $h = 12$, the number of subsets to be considered is 125970, and we need approximately 3 hours to enumerate all subsets (using the fastest algorithm available to find the MVEE for each subset) on a desktop computer. For $m = 30$, $n = 2$, and $h = 17$, approximately 43 days is necessary to enumerate all subsets.

There are various algorithms in the literature that address the problem of finding the MVE estimator. These algorithms can be divided into three categories: Exact methods, heuristics, and metaheuristics. Cook, Hawkins, and Weisberg [5] proposed a direct enumeration algorithm in which all h-point subsets are investigated. This method is very slow and can only be useful for very small instances as expected. Later, Agullo [1] proposed a branch and bound (B-and-B) algorithm which avoids investigating a large proportion of the subsets. We will discuss details of this algorithm in Section 5.3. Rousseeuw and Leroy [26] suggested a random search in which a number of subsets with $n + 1$ points are generated. The MVEE of these subsets are calculated and then deflated or inflated to enclose at least h points. The smallest of these ellipsoids is chosen as the MVE estimator. This is a rough estimate and the number of subsets that are investigated should be very large in order to have a good approximation. Hawkins [12] suggested the Feasible Solution Algorithm (FSA) which starts with a random h-point subset and searches the neighborhood by swaping two points each time, i.e., one point in the current subset is exchanged with another point that is not in the current subset, until no further improvement is possible by two-way swapping. It is discussed that the algorithm should make approximately 5000 random starts to find a satisfactory estimate. Modified versions of the FSA algorihm are also discussed which use different methods to choose the swaps. Also a lower bound is used to eliminate some subsets before

calculating the minimum-volume enclosing ellipsoid. Hawkins and Olive [13] improved these algorithms by using an easy-to-check condition at the initialization: eliminating some of the covered points if the initial sample covers more than h points. Poston [21] proposed an algorithm based on EID values. The EID value of a point is an estimate of the contribution of the point to the determinant of the Fischer information matrix which is closely related to the volume of the minimum-volume enclosing ellipsoid as discussed in Poston and Tolson [20]. This algorithm is very fast but can be far away from the optimal solution. Hawkins' FSA algorithms and Poston's EID algorithm are combined by Grambow and Stromberg [9] to obtain better approximations. Woodruff and Rocke [34] applied various metaheuristics such as genetic algorithms, tabu search, and simulated annealing to the MVE problem.

In this chapter, we will develop a 2-exchange heuristic. A 2-exchange or 2-opt heuristic is a local search method which produces a solution that is better than every other solution that can be obtained by dropping one point from the current subset and adding another one. Hawkin's FSA algorithm is also a 2-exchange heuristic; nevertheless our algorithm uses stronger bounds and various initializations. We will also develop a B-and-B algorithm and provide computational results. This algorithm is similar to Agullo's but uses different branching strategies and stronger bounds.

Although our results improve the older ones to some extent and provide valuable insight (mostly gained from our journey documented in this thesis), it is not a panacea. Exactly solving large instances of this problem requires a B-and-B algorithm developed for multiple processors and stronger bounds.

5.1 Preliminaries

In this chapter, we will assume that the data set $\mathcal{X}$ and all subsets of $\mathcal{X}$ being evaluated are centered around the origin. This is without loss of generality, since given an arbitrary subset we can obtain an equivalent problem on a centered subset as discussed in Chapter 2. Let $\mathcal{A} \subseteq \mathcal{I}$ be a set of indices. Let us recall the MVEE problem and its dual for the set $\mathcal{X}_{\mathcal{A}} := \{x_i \in \mathcal{X} : i \in \mathcal{A}\} \subseteq \mathcal{X}$. The primal problem is written as

$$
\begin{array}{rrcl}
 & \min_{H \succ 0} \quad f_{\mathcal{A}}(H) & := & -\ln\det H \\
(\mathcal{P}_{\mathcal{A}}) & x_i^T H x_i & \leq & n,\ i \in \mathcal{A},
\end{array}
$$

and its dual as

$$
\begin{array}{rrcl}
 & \max_u \quad g_{\mathcal{A}}(u) & := & \ln\det XUX^T \\
(\mathcal{D}_{\mathcal{A}}) & e^T u & = & 1, \\
 & u & \geq & 0, \\
 & u_i & = & 0,\ i \notin \mathcal{A},
\end{array}
$$

where $X = [x_1, \ldots, x_m]$ as in the previous chapters.

Definition 5.1.1 *A feasible solution for the MVE estimator problem is any subset $\mathcal{A} \subset \mathcal{I}$ such that $|\mathcal{A}| \geq h$. Its value is equal to $f_{\mathcal{A}}(H^*)$, where H^* is the optimal solution for $(\mathcal{P}_{\mathcal{A}})$.*

We need to solve $(\mathcal{P}_{\mathcal{A}})$ and $(\mathcal{D}_{\mathcal{A}})$ in order to evaluate the value of each subset. When two subsets lie in the neighborhood of each other in the search space, (i.e., one subset can be obtained by adding or removing a single point, or exchanging two points), solving the MVEE problem for one of them gives information about the value of the other one as discussed next. For simplicity we will assume that

the data points are indexed so that $\mathcal{A} = \{1, 2, \ldots, |\mathcal{A}|\}$. We also assume that we obtain new subsets with adding $|\mathcal{A}| + 1$, or removing $|\mathcal{A}|$, or doing both at the same time.

Lemma 5.1.1 *Given $\mathcal{A} = \{1, \ldots, k-1\}$ and $\hat{\mathcal{A}} = \mathcal{A} \cup \{k\}$, let H, $\hat{H}$, u, and $\hat{u}$ be optimal for $(\mathcal{P}_{\mathcal{A}})$, $(\mathcal{P}_{\hat{\mathcal{A}}})$, $(\mathcal{D}_{\mathcal{A}})$, and $(\mathcal{D}_{\hat{\mathcal{A}}})$, respectively. If $\xi_k \leq n$, then*

$$f_{\hat{\mathcal{A}}}(\hat{H}) = f_{\mathcal{A}}(H).$$

Otherwise,

$$f_{\hat{\mathcal{A}}}(\hat{H}) - f_{\mathcal{A}}(H) \geq (n-1)\log(1-\tau) + \log(1-\tau+\tau\xi_k), \tag{5.1.1}$$

where $\tau = \frac{\xi_k/n-1}{\xi_k-1}$ and $\xi_k = x_k^T(XUX^T)^{-1}x_k$.

Proof: When $\xi_k \leq n$, x_k is already enclosed by MVEE($\mathcal{X}_{\mathcal{A}}$); hence MVEE($\mathcal{X}_{\mathcal{A}}$) is also MVEE($\mathcal{X}_{\hat{\mathcal{A}}}$).

Assume $\xi_k > n$. For any $0 \leq \tau < 1$, $u_+ := (1-\tau)u + \tau e_k$ is feasible w.r.t $(\mathcal{D}_{\hat{\mathcal{A}}})$, where e_k is the k^{th} unit vector. Using (2.1.4), we have

$$\begin{aligned} f_{\hat{\mathcal{A}}}(\hat{H}) = g_{\hat{\mathcal{A}}}(\hat{u}) \geq g_{\hat{\mathcal{A}}}(u_+) &= (n-1)\log(1-\tau) + \log(1-\tau+\tau\xi_k(u)) + g_{\mathcal{A}}(u) \\ &= (n-1)\log(1-\tau) + \log(1-\tau+\tau\xi_k(u)) + f_{\mathcal{A}}(H). \end{aligned}$$

The rest of the proof follows from the fact that $\tau = \frac{\xi_k/n-1}{\xi_k-1}$ maximizes the right-hand side of this inequality. □

Lemma 5.1.2 *Given $\mathcal{A} = \{1, \ldots, k\}$ and $\hat{\mathcal{A}} = \mathcal{A} - \{k\}$, let H, $\hat{H}$, u, and $\hat{u}$ be optimal for $(\mathcal{P}_{\mathcal{A}})$, $(\mathcal{P}_{\hat{\mathcal{A}}})$, $(\mathcal{D}_{\mathcal{A}})$, and $(\mathcal{D}_{\hat{\mathcal{A}}})$, respectively. If $u_k = 0$, then*

$$f_{\hat{\mathcal{A}}}(\hat{H}) = f_{\mathcal{A}}(H).$$

Otherwise,

$$f_{\hat{\mathcal{A}}}(\hat{H}) - f_{\mathcal{A}}(H) \geq -n\log(1-u_k) + \log(1-nu_k).$$

Proof: When $u_k = 0$, u is feasible and also optimal for $(\mathcal{D}_{\hat{\mathcal{A}}})$; hence MVEE$(\mathcal{X}_{\mathcal{A}})$ is also MVEE$(\mathcal{X}_{\hat{\mathcal{A}}})$.

Suppose now that $u_k > 0$. Since u is optimal w.r.t $(\mathcal{D}_{\mathcal{A}})$, $u_k > 0$ implies that $\xi_k = n$. Consider the update $u_+ := (1-\tau)u + \tau e_k$ where $\tau = \frac{-u_k}{1-u_k}$. Since $(u_+)_k = 0$, u_+ is feasible w.r.t $(\mathcal{D}_{\hat{\mathcal{A}}})$. Using (2.1.4), we have

$$\begin{aligned} f_{\hat{\mathcal{A}}}(\hat{H}) = g_{\hat{\mathcal{A}}}(\hat{u}) &\geq g_{\hat{\mathcal{A}}}(u_+) = (n-1)\log(1/(1-u_k)) + \log((1-\xi_k u_k)/(1-u_k) + g_{\mathcal{A}}(u) \\ &= -n\log(1-u_k) + \log(1-nu_k) + f_{\mathcal{A}}(H). \end{aligned}$$

□

Assuming that we have found the MVEE of a particular subset $\mathcal{X}_{\mathcal{A}}$, Lemmas 5.1.1 and 5.1.2 provide lower bounds on the value of the subsets that can be obtained by adding an element to $\mathcal{A}$ or removing one from $\mathcal{A}$. These lower bounds will be useful to eliminate some subsets even without solving the MVEE problem for them. The following lemma takes a further step ahead and derives a lower bound on the value of the subsets that can be obtained by adding a point and removing another one from the current subset $\mathcal{A}$ simultaneously. This will be useful in developing the 2-exchange heuristic in the following section.

Lemma 5.1.3 *Given two sets $\mathcal{A} = \{1, \ldots, k-1, k\}$ and $\hat{\mathcal{A}} = \{1, \ldots, k-1, k+1\}$, let H, $\hat{H}$, u, and $\hat{u}$ be optimal for $(\mathcal{P}_{\mathcal{A}})$, $(\mathcal{P}_{\hat{\mathcal{A}}})$, $(\mathcal{D}_{\mathcal{A}})$, and $(\mathcal{D}_{\hat{\mathcal{A}}})$, respectively. Then we have*

$$f_{\hat{\mathcal{A}}}(\hat{H}) - f_{\mathcal{A}}(H) \geq (n-1)\ln(1-\tau_2) - n\ln(1-u_k) + \ln(1-u_k\xi_k + \tau_2\delta),$$

where $\delta = -1 + u_k\xi_k + (1-u_k)\xi_{k+1} - u_k(1-u_k)\left(\xi_k\xi_{k+1} - \xi_{k,k+1}^2\right)$ *and* $\tau_2 = \max\{0, \frac{\delta - n + nu_k\xi_k + 1 - u_k\xi_k}{n\delta}\}$.

Proof: Consider

$$u_+ := (1 - \tau_1 - \tau_2)u + \tau_1 e_k + \tau_2 e_{k+1}. \tag{5.1.2}$$

If we choose

$$\tau_1 = \frac{u_k(\tau_2 - 1)}{1 - u_k}, \tag{5.1.3}$$

then $(u_+)_k = 0$. Then u_+ is a feasible solution for $(\mathcal{D}_{\hat{\mathcal{A}}})$. We can calculate the dual objective function value at this point easily as follows. We have

$$XU_+X^T = (1 - \tau_1 - \tau_2)(XUX^T) + \tau_1 x_k x_k^T + \tau_2 x_{k+1} x_{k+1}^T,$$

which leads to

$$\begin{aligned}\det(XU_+X^T) = \det(XUX^T)(1 - \tau_1 - \tau_2)^n(1 + \frac{\tau_1}{1 - \tau_1 - \tau_2}\xi_k + \\ \frac{\tau_2}{1 - \tau_1 - \tau_2}\xi_{k+1} + \frac{\tau_1\tau_2}{(1 - \tau_1 - \tau_2)^2}(\xi_k\xi_{k+1} - \xi_{k,k+1}^2)),\end{aligned} \tag{5.1.4}$$

where $\xi_{k(k+1)} := x_k(XUX^T)^{-1}x_{k+1}$. Plugging in (5.1.3) and taking logarithms, we get

$$\ln\det(X\hat{U}_+X^T) = \ln\det(XUX^T) + (n-1)\ln(1 - \tau_2) - n\ln(1 - u_k) + \ln(1 - u_k\xi_k + \tau_2\delta).$$

The rest of the proof follows since $\tau_2 = \max\{0, \frac{\delta - n + nu_k\xi_k + 1 - u_k\xi_k}{n\delta}\}$ maximizes the right-hand side of this inequality while keeping u_+ feasible. □

Note that Lemma 5.1.3 deduces to Lemma 5.1.1 for $u_k = 0$ and to Lemma 5.1.2 for $\tau_2 = 0$ as expected.

5.2 A 2-Exchange Heuristic

In this section, we develop a 2-exchange heuristic which is capable of finding good solutions quickly even for large instances of the problem. The heuristic can roughly be summarized in three steps:

1. Find an initial feasible solution and evaluate its value.
2. Select an exchange that will improve the objective function.
3. Perform the exchange.

The last two steps are repeated until no improving exchange can be found. Since it is unlikely that a single run of the algorithm would return a local optimum, it should be repeated many times with many different initial solutions. As we will discuss below, the quality (by quality we mean the speed of the method as well as the quality of the solutions produced) of the heuristic is highly dependent on the initial solutions, the number of times the algorithm is called, and the selection of the points to be swapped at each iteration. Lower bounds on the value of the candidate subsets are useful in making clever decisions so that the number of swaps is reduced. Running time can further be reduced by careful implementation especially when performing the exchange in Step 3. We will elaborate on these details in the following subsections.

5.2.1 Finding a feasible initial solution

Here we introduce several methods to find an initial feasible solution. While some of the methods are elegant but computationally expensive, others are crude but cheap.

a. Ellipsoidal peeling

Ellipsoidal peeling starts with the MVEE of the whole data set $\mathcal{X}$ and 'peels' this ellipsoid by discarding one of the points on its boundary at each iteration until only h points remain. Lemma 5.1.2 provides an estimate (lower bound)

on the volume of the new MVEE when one of the points is discarded from the current subset. Since the lower bound is a decreasing function of the weight of the point being discarded, a point on the boundary with maximum weight is left outside. This methods starts with covering all points and generates ellipsoids which cover $m-1$, $m-2$, ..., and h points as it proceeds. It can be summarized as follows:

Algorithm 5.2.1 (Ellipsoidal Peeling)

Input: $X \in \mathbb{R}^{n\times m}$, $h \in \{1,\ldots,m\}$, *and* $\epsilon > 0$.

Step 1. *Set* $\mathcal{A} = \mathcal{I}$.

Step 2. If $|\mathcal{A}| = h$, ***STOP:*** *Output* $\mathcal{A}$.

Else, *run WA-TY to obtain an* ϵ*-approximate optimal solution* u *for* $(\mathcal{D}_{\mathcal{A}})$.

Step 3. *Find* $j = \arg\max_i u_i$. *Set* $\mathcal{A} = \mathcal{A} - \{j\}$. ***Go to*** *Step 2.*

Although ellipsoidal peeling usually generates good solutions, it has two major disadvantages. First, it is an expensive algorithm because we need to solve $m-h$ instances of the MVEE problem. Second, it is a deterministic algorithm which is not useful in designing a local search heuristic.

b. Ellipsoidal ordering

The following algorithm calculates the MVEE of the whole data set, orders points with respect to their ellipsoidal distance (defined using the MVEE), and picks the points with the h smallest distances as the initial solution.

Algorithm 5.2.2 (Ellipsoidal Ordering)

Input: $X \in \mathbb{R}^{n\times m}$, $h \in \{1,\ldots,m\}$, *and* $\epsilon > 0$.

Step 1. *Run WA-TY to obtain an* ϵ*-approximate optimal solution* u *for* $(\mathcal{D}_{\mathcal{A}})$.

Step 2. *Sort* $\xi_{i_1}(u) \le \xi_{i_2}(u) \le \ldots \le \xi_{i_m}(u)$. *Output* $\mathcal{A} = \{i_1,\ldots,i_h\}$.

c. The EID Algorithm

The EID algorithm discussed in [21] is a quick way to find a feasible solution which may provide good solutions for some problems. We provide the details of the algorithm in the following table for completeness. Note that $X(:,\mathcal{A})$ is a matrix whose columns are vectors in $\mathcal{X}_{\mathcal{A}}$.

Algorithm 5.2.3 (The EID Algorithm)

Input: $X \in \mathbb{R}^{n\times m}$ *and* $h \in \{1,\ldots,m\}$.

Step 1. *Let* $\mathcal{A} := \{1,\ldots,m\}$.

Step 2. *Calculate* $\bar{x} = \frac{\sum_{i\in\mathcal{A}} x_i}{|\mathcal{A}|}$ *and* $\bar{X} = X(:,\mathcal{A}) - [\bar{x},\ldots,\bar{x}]$.

Step 3. *Calculate* $\mathrm{EID} = \mathrm{diag}(X^T(\bar{X}\bar{X}^T)^{-1}X)$. *Let* $j = \arg\max_{\{i\in\mathcal{A}\}} \mathrm{EID}_i$.

Update $\mathcal{A} = \mathcal{A} - \{j\}$.

Step 4. If $|\mathcal{A}| = h$, ***STOP:*** *Output* $\mathcal{A}$.

Else, go to *Step 2.*

The EID values represent the contribution of the i^{th} point to the eigenvalues of the information matrix (see Chapter 1). Since the shape matrix of the MVEE

is related to the information matrix, removing the points with large EID values yields a solution with reasonably small volume. We need to be careful when implementing this algorithm so that the matrix X used in the algorithm does not lose its full rank property. This can be managed easily, since it is shown in [21] that removing any point x_i with $\text{EID}_i = 1$ leads to a singular matrix X. If this happens, we can instead remove the point with the second (or third, etc.) highest EID value.

d. Spherical ordering

Another idea is to order points with respect to their Euclidean norms and pick a subset with h smallest points as the initial solution.

Algorithm 5.2.4 (Spherical Ordering)

Input: *$X \in \mathbb{R}^{n\times m}$ and $h \in \{1, \ldots, m\}$.*

Step 1. *Sort $\|x_{i_1}\|_2 \leq \|x_{i_2}\|_2 \leq \ldots \leq \|x_{i_m}\|_2$. Output $\mathcal{A} = \{i_1, \ldots, i_h\}$.*

e. Random ellipsoidal peeling/building

The following algorithm randomizes the ellipsoidal peeling algorithm. It is useful in local search and usually very fast compared to its deterministic version.

Algorithm 5.2.5 (Random Ellipsoidal Peeling/Building)

Input: $X \in \mathbb{R}^{n\times m}$, $h \in \{1,\ldots,m\}$, *and* $\epsilon > 0$.

Step 1. *Choose a subset* $\mathcal{A}$ *randomly s.t.* $\mathcal{A} \subset \mathcal{I}$, $|\mathcal{A}| = n+1$, *and* $\mathcal{X}_{\mathcal{A}}$ *spans* $\mathbb{R}^n$.

Step 2. *Run WA-TY to obtain an* ϵ*-approximate optimal solution* u *for* $(\mathcal{D}_{\mathcal{A}})$.

Step 3. *Let* $\mathcal{A} = \{i : \xi(u) \le n(1+\epsilon)]\}$.

Step 4. If $|\mathcal{A}| = h$, ***STOP:*** *Output* $\mathcal{A}$.

elseif $|\mathcal{A}| < h$, *set* $\mathcal{A} = \mathcal{A} \cup \{j\}$, *where* $j = \arg\min\{\xi_i(u) : \xi_i(u) > n(1+\epsilon)\}$;

else *set* $\mathcal{A} = \mathcal{A} - \{j\}$, *where* $j = \arg\max_i u_i$.

Step 5. Go to *Step 2.*

f. Random ellipsoidal ordering

Similarly, we can also randomize the ellipsoidal ordering algorithm as follows.

Algorithm 5.2.6 (Random Ellipsoidal Ordering)

Input: $X \in \mathbb{R}^{n\times m}$, $h \in \{1,\ldots,m\}$, *and* $\epsilon > 0$.

Step 1. *Choose a subset* $\mathcal{A}$ *randomly s.t.* $\mathcal{A} \subset \mathcal{I}$, $|\mathcal{A}| = (n+1)$, *and* $\mathcal{X}_{\mathcal{A}}$ *spans* $\mathbb{R}^n$.

Step 2. *Run WA-TY to obtain an* ϵ*-approximate optimal solution* u *for* $(\mathcal{D}_{\mathcal{A}})$.

Step 3. *Sort* $\xi_{i_1}(u) \le \xi_{i_2}(u) \le \ldots \le \xi_{i_m}(u)$.

Output $\mathcal{A} = \{i_1,\ldots,i_h\}$.

5.2.2 Selecting points for the exchange

At each iteration of the heuristic, we move from the current solution (a subset $\mathcal{A}$ with h points) to a new one in its neighborhood by making an exchange between a covered point $i \in \mathcal{A}$ and an uncovered point $j \notin \mathcal{A}$. Although there are $h \times (m - h)$ possible exchanges, considering exchanges between points on the boundary of the MVEE of the current subset and the $(m - h)$ uncovered ones is enough because exchanging a point in the interior of the current ellipsoid will not lead to a subset whose MVEE has smaller volume. Since there are $O(n)$ points on the boundary of an MVEE in practice, the number of possible exchanges decreases dramatically with this basic observation. For each candidate pair of points to be exchanged, the lower bound (on the volume of the candidate subset) from Lemma 5.1.3 is used in order to eliminate nonpromising exchanges. Once we obtain a candidate exchange x_i and x_j whose lower bound is lower than the current objective function value, we solve the corresponding MVEE problem for the candidate set $\hat{\mathcal{A}} = \mathcal{A} - \{i\} \cup \{j\}$. If the volume of MVEE($\hat{\mathcal{A}}$) is smaller than that of MVEE($\mathcal{A}$), we record this value. Three different versions of the algorithm can be used:

i. BEST EXCHANGE version: evaluates all possible exchanges and uses the best one.

ii. FIRST EXCHANGE version: picks the first exchange that provides an MVEE with a smaller value.

iii. BEST BOUND version: calculates a lower bound (from Lemma 5.1.3) for all possible exchanges, picks the best, and calculates its value. This exchange is used if the value of the candidate solution is better than the

value of the current solution, otherwise the exchange with the second (or third, etc.) smallest lower bound is considered.

All versions of the heuristic are faster than the previously reported heuristics in the literature because of the decreased number of exchanges considered at every iteration and the improvement on the lower bound. In addition, we are capable of evaluating each candidate exchange faster because we don't solve the corresponding MVEE problem from scratch but start at a feasible solution obtained from the current optimal solution for MVEE($\mathcal{A}$) using the best step size as discussed in Lemma 5.1.3. In addition, we stop evaluating (running the WA-TY algorithm) a subset when the objective function value exceeds the best solution found so far.

We can summarize the heuristic as follows:

Algorithm 5.2.7 (2-Exchange Heuristic: BEST EXCHANGE)

Input: *$X \in \mathbb{R}^{n \times m}$, $h \in \{1, \ldots, m\}$, and $\epsilon > 0$.*

Step 1. *Let $\mathcal{A}$ be the output from one of the Algorithms 5.2.1-5.2.6.*

Let H and u be ϵ-approximate optimal solutions for $(\mathcal{P}_{\mathcal{A}})$ and $(\mathcal{D}_{\mathcal{A}})$.

Set $Bvol = f_{\mathcal{A}}(H)$ and $opt = 1$.

Step 2. *For all $k : u_k > 0$ and all $j : j \notin \mathcal{A}$, calculate*

$$LB_{kj} = f_{\mathcal{A}}(H) + (n-1)\ln(1-\tau_2) - n\ln(1-u_k) + \ln(1 - u_k\xi_k(u) + \tau_2\delta),$$

where τ_2 and δ are as in Lemma 5.1.3 with j replacing $k+1$,

and let $\hat{\mathcal{A}} = \mathcal{A} - \{k\} \cup j$.

If *$LB_{kj} < Bvol$,*

find $\hat{H}$ and $\hat{u}$ ϵ-approximate optimal solutions for $(\mathcal{P}_{\hat{\mathcal{A}}})$ and $(\mathcal{D}_{\hat{\mathcal{A}}})$.

If *$f_{\hat{\mathcal{A}}}(\hat{H}) < Bvol$, $Bvol = f_{\hat{\mathcal{A}}}(\hat{H})$, $c\mathcal{A} = \hat{\mathcal{A}}$, $cH = \hat{H}$, $cu = \hat{u}$ and $opt = 0$.*

Step 3. If *$opt = 0$, then set $\mathcal{A} = c\mathcal{A}$, $H = cH$, $u = cu$ and $opt = 1$.* ***Go to*** *Step 2.*

Else*, output Bvol and $\mathcal{A}$.*

5.3 A Branch-and-Bound Algorithm

We now present a branch and bound (B-and-B) algorithm in order to obtain exact solutions for small- and medium-sized data sets. The B-and-B algorithm starts with calling the heuristic and obtaining an upper bound on the volume of the MVE estimator. Then all n point (ordered lexicographically) subsets of the set $\mathcal{I}$ are generated and put into a stack. Each subset in the stack corresponds to a subproblem to be solved. We process the stack in FIFO (first in first out) order.

At each step of the algorithm, we solve the MVEE problem on the subset S at hand (the first subproblem on the stack) and obtain a dual optimal solution u. The WA-TY algorithm is called anytime a subproblem is to be solved as a black-box. If the volume of the MVEE(S) is greater than the upper bound, we fathom this subproblem by erasing it from the stack, since the volume of the MVEE(S) is a lower bound on the volume of the MVEE of each subset which contains S. Otherwise, if the number of points in the subset is h, the upper bound is updated to be the volume of the MVEE(S) and the problem is again fathomed. On the other hand, if the volume of MVEE(S) is smaller than the upper bound and the number of points in S is smaller than h, we branch on the subproblem by generating new subproblems. Each new subproblem corresponds to a subset S' which is generated by adding a new point to the current subset S. (Since we add only points with indices larger than the last element in the ordered set S, each subset of $\mathcal{I}$ is considered at most once with this branching rule.) Using Lemma 5.1.1, the optimal solution for the MVEE problem for the subset S provides a lower bound on the volume of the MVEE of the subset S'. If the lower bound for S' is smaller than the upper bound, the new subproblem S' is added to the end of the stack. Furthermore, the efficiency of the B-and-B algorithm can be increased by storing the solution $(1-\tau)u + \tau e_j$ (with τ as in Lemma 5.1.1) for each subproblem $S' = S \cup \{j\}$ since the blackbox for the MVEE problem will solve the problem for S' more efficiently using this solution. It is important to note that some of the subproblems are solved with no further effort as soon as they are generated. Let $S' = S \cup \{j\}$. It is obvious that, if $\xi_j \leq n$, x_j is covered by MVEE(S) and hence MVEE(S')=MVEE(S) and the solution u is optimal for both problems. Therefore, we can continue branching on S' at no cost. We will refer to this as aggressive branching.

The B-and-B algorithm is similar to that of Agullo [1]. Nevertheless, we make use of tighter lower bounds on the subproblems, a better upper bound obtained from the heuristic, a faster routine to solve the MVEE problems on subproblems, and also agressive branching. We are capable of solving small instances of the problem very fast, and medium-sized problems in a couple of hours as discussed in the next section. Although we expect our algorithm to be faster than that of Agullo [1], we cannot make accurate comparisons since the solution times are highly dependent on the data set X. Experiments on a wide range of data sets should be performed on the same platform for a fair comparison. This is not practical at this moment since Agullo's algorithm has been coded in FORTRAN (which is faster in data storage and movement than MATLAB) and is not available to the author.

5.4 Computational Study

In this section, we provide a computational study comparing the BEST EXCHANGE version of the 2-exchange heuristic and the B-and-B algorithm. We have coded both algorithms in MATLAB using the WA-TY algorithm with elimination technique from Chapter 2 as the black box MVEE solver whenever needed. The heuristic makes 204 random starts, one for each deterministic method (i.e., Algorithms 5.2.1-5.2.4) and 100 for each non-deterministic method (i.e., Algorithms 5.2.5-5.2.6). The subset with the smallest objective function value is chosen as the MVE estimator by the heuristic and its value is also used as an initial upper bound for the B-and-B algorithm.

We have started by conducting experiments on the six classical data sets that

had been used by various authors in the literature. The parameters of these data sets together with their names are given in Table 5.1. The actual data sets are available in [26].

Table 5.1: Parameters for Data Sets used in the Literature

Data Set	n	m	h
Aircraft	4	23	14
Coleman	5	20	13
Delivery	2	25	14
Education	3	50	27
Gravity	5	20	13
Salinity	3	28	16

Table 5.2: Solutions Times for the Heuristic and the B-and-B Algorithm for the Classical Data Sets

Data Set	Heuristic time (sec.)	B-and-B time (sec.)	Decrease (%)
Aircraft	15.6	14.5	0
Coleman	11.4	3.5	0
Delivery	10.2	1.8	0
Education	53	6186	0
Gravity	8.7	2.7	0
Salinity	19.2	9.3	0

We have used a 3.40 GHz Pentium IV processor with 1.0 GB RAM using MATLAB version R2006b and set $\epsilon = 10^{-2}$. The run times of the heuristic and the exact algorithm for each data set are given in Table 5.2. It is not surprising

Table 5.3: Average Solutions Times for the Heuristic and the B-and-B Algorithm for Data Sets with Standard Normal Distribution

Dimensions			Heuristic	B-and-B	No. of	Max. %
n	m	h	time	time	Unsolved	Decrease
2	20	12	3.3	0.2	2	0.6
2	30	18	5.4	1.3	2	0.4
2	50	27	35.2	56.9	7	2
3	20	12	10.3	1.1	2	8.4
3	30	18	20.1	10.9	5	0.5
3	50	27	45.8	697.2	6	0.1
5	20	12	16.2	5.0	2	0.1
5	30	18	38.8	119.9	2	0.1
5	50	27	108.2	3205.2	8	2.3

that the exact algorithm is faster than the heuristic for small instances, since the number of branches (i.e., MVEE problem instances) that need to be solved in the B-and-B tree is very small, while the heuristic makes many starts and hence solves many MVEE problems even for small data sets. The last column represents the percent decrease in volume by the B-and-B algorithm. Each entry in this column is zero indicating that the heuristic was able to find the optimal solution for all of the data sets. We will see that this is not true for other data sets; nevertheless the gap is quite small.

We have tested the algorithms on normally distributed data sets with parameters varying from $(n, m) = (2, 20)$ to $(n, m) = (5, 50)$. For each set of parameters, 100 instances were tested and the results are presented in Table 5.3. The first

three columns demonstrate the parameters (dimensions). The next two columns give the corresponding average solution times for the heuristic and the B-and-B algorithms. The column labeled with 'No. of Unsolved' provides the number of cases out of 100 that were not solved to optimality by the heuristic and the last column provides the worst optimality gap in percent among these unsolved cases. We are able to solve small- to medium-sized problems in a reasonable amount of time and the heuristic provides good solutions in under two minutes for all instances. We are also able to obtain solutions in under five minutes for parameters as large as $(n, m, h) = (5, 400, 203)$ using the 2-exchange heuristic.

BIBLIOGRAPHY

[1] J. Agullo. Exact iterative computation of the multivariate minimum volume ellipsoid estimator with a branch and bound algorithm. In A. Prat, editor, *Proceedings in Computational Statistics*, pages 175–180. Heidelberg: Physica-Verlag, 1996.

[2] S. D. Ahipaşaoğlu, P. Sun, and M. J. Todd. Linear convergence of a modified Frank-Wolfe algorithm for computing minimum volume enclosing ellipsoids. *Optimization Methods and Software*, 23:5–19, 2008.

[3] C. L. Atwood. Optimal and efficient designs of experiments. *The Annals of Mathematical Statistics*, 40:1570–1602, 1969.

[4] C. L. Atwood. Sequences converging to D-optimal designs of experiments. *The Annals of Statistics*, 1:342–352, 1973.

[5] R. D. Cook, D. M. Hawkins, and S. Weisberg. Exact iterative computation of the robust multivariate minimum volume ellipsoid estimator. *Statistics and Probability Letters*, 16:213–218, 1993.

[6] A.L. Dontchev and R.T. Rockafellar. Characterizations of Lipschitzian stability in nonlinear programming. In A.V. Fiacco, editor, *Mathematical Programming with Data Perturbations*, pages 65–82. New York: Marcel Dekker, Inc., 1998.

[7] V. V. Fedorov. *Theory of Optimal Experiments*. Academic Press, New York, 1972.

[8] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Nav. Res. Log. Quart.*, 3:95–110, 1956.

[9] S. C. Grambow and A. J. Stromberg. Combining the EID and FSA for computing minimum volume ellipsoid. Technical report, Dept. of Stats., University of Kentucky, 1998.

[10] J. Guélat and P. Marcotte. Some comments on Wolfe's 'away step'. *Mathematical Programming*, 35:110–119, 1986.

[11] R. Harman and L. Pronzato. Improvements on removing non-optimal support points in D-optimum design algorithms. *Statistics and Probability Letters*, 77:90–94, 2007.

[12] D. M. Hawkins. A feasible solution for the minimum volume ellipsoid estimator in multivariate data. *Computational Statistics*, 8:95–107, 1993.

[13] D. M. Hawkins and D. J. Olive. Improved feasible solution algorithms for high breakdown estimation. *Computational Statistics and Data Analysis*, 30:1–11, 1999.

[14] St. R. C. John and N. R. Draper. D-optimality for regression designs: A review. *Technometrics*, 17:15–23, 1975.

[15] L. G. Khachiyan. Rounding of polytopes in the real number model of computation. *Mathematics of Operations Research*, 21:307–320, 1996.

[16] L. G. Khachiyan and M. J. Todd. On the complexity of approximating the maximal inscribed ellipsoid for a polytope. *Mathematical Programming*, 61:137–159, 1993.

[17] J. Kiefer and J. Wolfowitz. The equivalence of two extremum problems. *Can. J. Math.*, 12:363–366, 1960.

[18] P. Kumar and E. A. Yıldırım. Minimum volume enclosing ellipsoids and core sets. *J. Optimization Theory and Applications*, 126(1):1–21, 2005.

[19] H. W. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

[20] W. L. Poston and R. H.Tolson. Maximizing the determinant of the information matrix with the effective independence distribution method. *AIAA Journal of Guidance, Control and Dynamics*, 15:1513–1514, 1992.

[21] W. L. Poston, E. J. Wegman, C. E. Priebe, and J. L. Solka. A deterministic method for robust estimation of multivariate location and shape. *Journal of Computational and Graphical Statistics*, 6:300–313, 1997.

[22] F. Pukelsheim. *Optimal Design of Experiments*. John Wiley and Sons, Inc., New York, 1993.

[23] E. Rimon and S. P. Boyd. Obstacle collision detection using best ellipsoid fit. *Journal of Intelligent and Robotic Systems*, 18:105–126, 1997.

[24] S. M. Robinson. Strongly regular generalized equations. *Math. Oper. Res.*, 5:43–62, 1980.

[25] S. M. Robinson. Generalized equations and their solutions, part II: Applications to nonlinear programming. *Math. Prog. Study*, 19:200–221, 1982.

[26] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. John Wiley and Sons, New York, 1987.

[27] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge University Press, 2004.

[28] N. Z. Shor. Cut-off method with space extension in convex programming problems. *Kibernetika*, 13(1):94–95, 1977. English translation: *Cybernetics* 13(1), 94–96.

[29] S. Silvey and D. Titterington. A geometric approach to optimal design theory. *Biometrika*, 62:21–32, 1973.

[30] S. D. Silvey. *Optimal Design: An Introduction to the Theory for Parameter Estimation*. Chapman and Hall, New York, 1980.

[31] P. Sun and R. M. Freund. Computation of minimum volume covering ellipsoids. *Operations Research*, 52:690–706, 2004.

[32] M. J. Todd and E. A. Yıldırım. On Khachiyan's algorithm for the computation of minimum volume enclosing ellipsoids. *Discrete and Applied Mathematics*, 155:1731–1744, 2007.

[33] P. Wolfe. Convergence theory in nonlinear programming. In J. Abadie, editor, *Integer and Nonlinear Programming*, pages 1–36. North-Holland, Amsterdam, 1970.

[34] D. L. Woodruff and D. R. Rocke. Heuristic search algoirhtms for the minimum volume ellipsoid. *Computational and Graphical Statistics*, 2:69–95, 1993.

[35] H. P. Wynn. The sequential generation of D-optimum experimental design. *Annals of Mathematical Statistics*, 41:1655–1664, 1970.

[36] H. P. Wynn. Results in the theory and construction of D-optimum experimental designs. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34:133–147, 1972.

[37] E. A. Yıldırım. Two algorithms for the minimum enclosing ball problem. *SIAM Journal on Optimization*, 19:1368–1391, 2008.

[38] D. B. Yudin and A. S. Nemirovskii. Informational complexity and efficient methods for the solution of convex extremal problems. *Ékonomika i Matematicheskie metody*, 12:357–369, 1976. English translation: *Matekon* 13(2), 3–25.