Technion – Israel Institute of Technology
Faculty of Industrial Engineering and Management
Minerva Optimization Center
Technion City, Haifa 32000, Israel
Fax 972-4-8235194

LECTURES ON MODERN CONVEX OPTIMIZATION

ANALYSIS, ALGORITHMS, ENGINEERING APPLICATIONS

Aharon Ben-Tal and Arkadi Nemirovski

Ben-Tal, A., and Nemirovski, A. *Lectures on Modern Convex Optimization: Analysis, Algorithms and Engineering Applications*, MPS-SIAM Series on Optimization, SIAM, Philadelphia, 2001

Preface

The goals. To make decisions optimally is a basic desire of a human being. Whenever the situation and the objectives can be described quantitatively, this desire can be satisfied, to some extent, by using mathematical tools, specifically those provided by Optimization theory and algorithms. For our purposes, a sufficiently general mathematical setting of an optimization problem is offered by *Mathematical Programming*:

$$\begin{array}{lrcl} \text{minimize} & f_0(x) & & \\ \text{subject to} & & & \\ & f_i(x) & \leq & b_i,\ i = 1, ..., m, \\ & x & \in & X \subset \mathbf{R}^n. \end{array} \tag{P}$$

In this problem, we are given an *objective* $f_0(x)$ and finitely many *functional constraints* $f_i(x)$, $i = 1, ..., m$, which are real-valued functions of n-dimensional *design vector* x varying in a given domain X. Our goal is to minimize the objective over the *feasible set* of the problem – the set which is cut off the domain X by the system of inequalities $f_i(x) \leq b_i$, $i = 1, ..., m$.

> In typical engineering applications, the design vector specifies a decision to be made (e.g., the physical sizes of the bars in a truss-like structure), the domain X is the set of "meaningful" values of the design vector, and the functional constraints represent design specifications – restrictions (physical, technical, financial) on certain characteristics of the decision.

The last decade has witnessed a major progress in Optimization, especially in the area of convex programming. Powerful modeling languages and database technology extended our abilities to model large scale complex real-life problems; progress in complexity theory improved our understanding of the advantages of certain algorithms, and the limitations of others, and resulted in developing efficient interior point algorithms for a wide family of convex programs. Combined with the dramatic improvement of computers, the above progress enables us today to solve problems which were considered out of reach for Optimization just ten years ago.

Regrettably, this promising state of affairs is yet unrealized, and consequently not utilized, by potential end-users (engineers, managers, etc.). This book aims at presenting modern optimization, combining it with applications (mainly from engineering) and thus helping to bridge the gap between researchers and practitioners. This ultimate goal determines the approach we are undertaking, and dictates several specific targets which we are trying to achieve.

Theoretically speaking, what modern Optimization can solve "well" are *convex optimization problems*. In essence, the two-decade-long investigation of complexity issues in Optimization can be summarized in the following conclusion:

> (!) *From the viewpoint of the numerical processing of problems* (P), *there exists a "solvable case" – the one of convex optimization problems, those where the domain X is a closed convex subset of $\mathbf{R}^n$, and the objective $f_0(x)$ and the functional constraints $f_i(x)$, $i = 1, ..., m$, are convex functions on X.*
>
> *Under minimal additional "computability assumptions"* (which are satisfied in basically all applications), *a convex optimization problem is "computationally tractable" – the computational effort required to solve the problem to a given accuracy "grows moderately" with the dimensions of the problem and the required number of accuracy digits.*

In contrast to this, general-type non-convex problems are too difficult for numerical solution; the computational effort required to solve such a problem, by the best numerical methods known so far, grows prohibitively fast with the dimensions of the problem and the number of accuracy digits. Moreover, there are serious theoretical reasons to conjecture that this is an intrinsic feature of non-convex problems rather than a drawback of the existing optimization techniques.

As an example, consider the pair of optimization problems (A) and (B) below. The first is the nonconvex problem

$$\begin{array}{lrcl} \text{maximize} & \sum_{i=1}^{n} x_i & & \\ \text{subject to} & & & \\ & x_i^2 - x_i & = & 0,\ i = 1, ..., n; \\ & x_i x_j & = & 0 \quad \forall (i,j) \in \Gamma, \end{array} \tag{A}$$

Γ being a given set of pairs (i, j) of indices i, j. This is a fundamental combinatorial problem of computing the *stability number* of a graph. It arises, e.g., in the following *channel communication problem*:

There is an alphabet of n letters a_i, $i = 1, 2, ..., n$, say, the 256 usual bytes. A letter a_i can be sent through a communication channel; when passing trough it, it either remains unchanged, or can be converted to another letter a_j due to transmission errors. The errors are assumed to be "symmetric" (if a_i can be converted into a_j, then a_j can be converted into a_i as well), and Γ is the set of (indices of) those pairs of letters which can be converted from one into another. Assume that we are interested in a "non-confusing" communication, where the addressee either gets a correct letter, or is able to conclude that a transmission error has occurred, but never misreads the input letter. In this case we should restrict the sub-alphabet S we use to be *independent*, meaning that no two distinct letters from S can be converted from one to another by the channel. In order to get from the channel "its most", we would like to use a independent sub-alphabet of maximal cardinality. It turns out that the optimal value in (A) is exactly the cardinality of such a maximal independent sub-alphabet.

The second problem is the convex program

$$\begin{array}{lrcl} & \text{minimize} & x_0 & \\ \text{subject to} & & & \\ \lambda_{\min} \begin{pmatrix} x_1 & & & x_1^\ell \\ & \ddots & & \vdots \\ & & x_m & x_m^\ell \\ x_1^\ell & \cdots & x_m^\ell & x_0 \end{pmatrix} & & \geq & 0,\ \ell = 1, ..., k, \\ & \sum_{j=1}^{m} a_j x_j^\ell & = & b^\ell,\ \ell = 1, ..., k, \\ & \sum_{j=1}^{m} x_j & = & 1, \end{array} \tag{B}$$

where $\lambda_{\min}(A)$ denotes the minimum eigenvalue of a symmetric matrix A. This problem originates from design of a *truss* (a mechanical construction built from thin elastic bars, like an electric mast, a bridge or the Eifel Tower) capable to withstand k nonsimultaneous loads.

Looking at the analytical forms of (A) and (B), it seems that the first problem is easier than the second: the constraints in (A) are simple explicit quadratic equations, while the constraints in (B) involve much more complicated functions

of the design variables – the eigenvalues of certain matrices depending on the design vector. The truth, however, is that the first problem is in a sense "as difficult as an optimization problem can be", and the worst-case computational effort to solve it within absolute inaccuracy 0.5 is about 2^n operations for all known optimization methods. For $n = 256$ ("alphabet of bytes"), the complexity $2^n \approx 10^{77}$ is, for all practical purposes, the same as $+\infty$. In contrast to this, the second problem is quite "computationally tractable". E.g., for $k = 6$ (6 loading scenarios) and $m = 100$ (a 100-bar construction) the problem has 701 variables (2.7 times the number of variables in the "byte" version of (A)); however, it can be reliably solved within 6 accuracy digits in a couple of minutes. The dramatic difference in the computational effort required to solve (A) and (B) is due to the fact that (A) is a non-convex optimization problem, while (B) is convex.

The above discussion explains the words "Convex Programming" in the title of our book. We now explain the word "modern". The emphasis in the book is on *well-structured convex problems* such as Linear, Conic Quadratic and Semidefinite Programming. These areas are the ones most affected by the recent progress in Optimization, areas where we possess well-developed techniques for building large-scale models, analyzing them theoretically ("on paper") and processing them numerically. Except for Linear Programming, these areas basically did not exist 10 years ago, so that most of the users who could potentially benefit from recent developments are not aware of their existence, let alone their usefulness. Enlarging the scope of classical optimization (LP, general Nonlinear Programming) by introduction of CQP and SDP, new application areas were created. Examples include: semidefinite relaxations of difficult combinatorial problems, Linear Matrix Inequality-based techniques in Control, Mathematical Programming with uncertain data, etc. These new applications create synergies between Optimization, Computer Science and Engineering, with potentially far-reaching consequences on our ability to solve complex real-life decision problems.

At this point, we want to make a remark addressing experts in optimization rather than "general" readers. The history of Convex Programming, as far as applied aspects are concerned, started with the invention of Linear Programming (G. Dantzig, circa 1948). LPs possess the simplest possible and transparent analytical structure, which from the very beginning was heavily exploited both theoretically (the completely algorithmic LP Duality Theory) and computationally (the Simplex method). With subsequent nonlinear extensions, the focus was shifted, basically "in one giant step", from the simplest possible linear structure to the most general one, where all we know about the entities occurring in (P) is that they are called f_i, $i = 0, 1, ..., m$, and X, that they are convex, and that f_i are $0/1/2/...$ times continuously differentiable. At the theoretical level, the focus on this general setting yielded very deep results in Convex Analysis (Lagrange duality, Karush-Kuhn-Tucker optimality conditions, etc.); however, "the price" of these really fundamental achievements was a lack of an *algorithmic* content of the subject. For example, the Lagrange dual of a general-type convex program (P) is something which exists and possesses very nice properties; this "something", however, normally cannot be written down explicitly (in a sharp contrast with the "fully algorithmic" LP duality). At the computational level, the "emphasis on generality" resulted in general-purpose "near-sighted" (and thus slow, as compared to LP algorithms) optimization methods, those utilizing purely local information on the problem.

To some extent, recent trends (the last decade or so) in Convex Optimization stem from realizing that there is something "in between" the relatively narrow Linear Programming and the completely unstructured universe of Convex Programming; this "something" are "well-structured" generic convex optimization problems like Conic Quadratic and Semidef-

inite Programming. Needless to say, interest in "special cases" is not a complete novelty for our area (recall Linearly Constrained Convex Quadratic and Geometric Programming); what is indeed a novelty, is the recent emphasis on well-structured generic problems, along with outstanding achievements resulting from this emphasis. The most remarkable of these achievements is, of course, the "interior point revolution", which has extended dramatically our abilities to process convex programs numerically, at the same time creating a completely new, challenging and attractive area of theoretical research. The emphasis on well-structured "special cases" has, however, another, a less evident (perhaps not less important) consequence which can be summarized as follows:

- *When restricted to "nice" generic convex programs, like LP, CQP and SDP, Convex Analysis becomes "an algorithmic calculus"* – basically as algorithmic as in the LP case. For example, the SDP Duality is as "explicit and symmetric" as the LP one. In fact, the same can be said about all other basic operations of Convex Analysis, from the simplest (like taking intersections and affine images of convex sets, or sums of convex functions) to the more sophisticated ones (like passing from a convex set to its polar or from a convex function to its Legendre transformation). Whenever the "operands" of such a construction can be represented, in a properly defined precise sense, via, say, SDP, the same is true for the resulting entity, and the "SDP representation" of the result is readily given by those of the operands.

  As a result,

- *An instance of a "nice" generic convex problem can be processed, up to a certain point, "on paper" (and in a quite routine fashion). In many cases this allows to obtain important qualitative information on the problem and/or to convert it into an equivalent one, better suited for subsequent numerical processing.* For example, applying SDP Duality, one can reduce dramatically the design dimension of the Truss Topology Design problem, and as a result – increase by orders of magnitude the sizes of the TTD problems which can be processed numerically in practice.

  Moreover,

- *"Nice" generic convex problems, like CQP and especially SDP, possess vast "expressive abilities"*, which allows to utilize the above advantages in a very wide spectrum of applications, much wider than the one covered by Linear Programming.

When writing this book, our major concern was to emphasize the issues just raised, and this emphasis is perhaps the most characteristic (and, hopefully, to some extent a novel) feature of the book.

Restricting ourselves to "well-structured" convex optimization problems, it becomes logical to skip a significant part of what is traditionally thought of as "the theory and algorithms of Mathematical Programming". Readers interested in the Gradient Descent, Quasi-Newton methods, and even Sequential Quadratic Programming, are kindly advised to use the excellent existing textbooks on these important subjects; our book should be thought of as a self-contained complement to, and not as an extended version of, these textbooks. We even have "dared" to omit the Karush-Kuhn-Tucker optimality conditions in their standard form, since they are too general to be "algorithmic"; the role of the KKT conditions in our exposition is played by their particular (and truly "algorithmic") case, expressed by the so called Conic Duality Theorem.

The book is primarily addressed to potential users (mainly, engineers). Consequently, *our emphasis is on building and processing instructive engineering models rather than on describing the details of optimization algorithms*. The underlying motivation is twofold. First, we are trying to convince an engineer that Optimization has indeed something to do with his/her

professional interests. Second, we believe that a crucial necessary condition for successful practical applications of optimization is the understanding of what is desirable and what should be avoided at the *modeling* stage. Thus, important questions to be addresses are:

(a) *What are optimization models which can be efficiently processed* (to certain point – "on paper" and then - on a computer),

(b) *How one can convert* (provided that it is possible) *a "seemingly bad" initial description of a problem into a "tractable and well-structured" optimization model.*

We believe that the best way to provide a reader with a relevant insight is to present, along with general concepts and techniques, many applications of these concepts and techniques. As about optimization algorithms, we believe that their presentation in a user-oriented book should be as non-technical as possible (to drive a car, no expertise in engines is necessary, otherwise there is something bad with cars...) The part of the book devoted to algorithms presents the Ellipsoid method (due to its simplicity, combined with its capability to answer affirmatively the fundamental question of whether Convex Programming is "computationally tractable") and an overview of polynomial time interior point methods for Linear, Conic Quadratic and Semidefinite Programming.

In spite of the fact that the book is user-oriented, it is a mathematical book. The goal we try to achieve is to demonstrate that when processing "meaningful" *mathematical* models by *rigorous* mathematical methods (and not by their "engineering surrogates"), one can obtain results which have meaningful and instructive engineering interpretation. Whether we have reached this goal – this is another story, and here the judgment is upon the reader.

Last, but not least, a reader should keep in mind that what follows are *Lecture Notes*; *our intention was to highlight those issues which we find most interesting and instructive*, rather than to present a complete overview of Convex Optimization. Consequently, we are ready to take blame for being boring, or unclear, or focusing on issues of a minor importance, in any material which we *include* in the book, however, we do not accept "money back" requests based on claims that something (however important) is *not* included. Along with "immunity w.r.t. what is absent", a "Lecture Notes type" book provides us with some other privileges, like a style which is a bit more vivid compared to the academic standards, and a rather short list of bibliography references (embedded as footnotes in the body of the text). In this latter respect, a reader should be aware that the fact that if a statement (even a "non-common-knowledge" one) appears in the text without a reference, this does *not* mean that we are claiming for the authorship; it merely reflects the fact that our focus is on the state of Convex Optimization rather than on its history.

Audience and prerequisites. Formally, readers are supposed to know basic facts from Linear Algebra and Analysis – those presented in standard undergraduate mathematical courses for engineers. As far as optimization-related areas are concerned, readers are assumed to know the definitions of a convex set and a convex function and to have heard (no more than that!) what is a Mathematical Programming problem. What is highly desirable, is a possession of the basic elements of mathematical culture.

The exercises accompanying each lecture form a significant part of the book. They are organized in small groups, each devoted to a specific topic somehow related to the corresponding lecture. Typically, the task in an exercise is to prove something. Most of the exercises are not too easy... The results stated by the exercises are used in the subsequent parts of the book in the same way as the statements presented and proved in the main body of the book; consequently,

a reader is kindly asked if not to do, then at least to read carefully all exercises. The order of exercises is of primary importance: in many cases preceding exercises contain information/hints necessary to succeed in the subsequent ones.

Acknowledgments. A quite significant part of applications discussed in our book is taken from the papers of Prof. Stephen Boyd from Stanford University and his colleagues. We are greatly indebted to Prof. Boyd for providing us with access to this material and stimulating discussions. Applications related to Structural Design were developed in tight collaboration with Prof. Jochem Zowe and Dr. Michael Kočvara from Erlangen University. Parts of the book were written when the authors were visiting the Statistics and Operations Research Department of the Technical University of Delft, and we are thankful to our hosts Prof. Kees Roos and Prof. Tamas Terlaky.

Aharon Ben-Tal and Arkadi Nemirovski

August 2000, Haifa, Israel

Contents