INTRODUCTION TO LINEAR AND NONLINEAR PROGRAMMING

DAVID G. LUENBERGER

INTRODUCTION TO LINEAR AND NONLINEAR PROGRAMMING

DAVID G. LUENBERGER
Stanford University

ADDISON-WESLEY PUBLISHING COMPANY
Reading, Massachusetts
Menlo Park, California · London · Don Mills, Ontario

Abstract design on cover: "Computer Composition With Lines,"
© A. Michael Noll 1965.

"Computer Composition With Lines," one of the earliest examples of computer-generated art, was inspired by the 1917 painting "Composition With Lines," by the Dutch artist Piet Mondrian. See A. Michael Noll, "Human or Machine: A Subjective Comparison of Piet Mondrian's 'Composition With Lines' (1917) and a Computer-Generated Picture," *The Psychological Record*, 16, 1966, pages 1-10.

ISBN 0-201-04347-5
ABCDEFGHIJ-MA-79876543

PREFACE

This book is intended as an introduction to a broad range of practical optimization techniques. It is designed for either self-study by professionals or classroom work at the undergraduate or graduate level for students who have a technical background in engineering, mathematics, or science. Over the past four years, the book has served, at various stages of its development, as the basis for several courses in optimization at Stanford University. Like the field of optimization itself which involves many classical disciplines—particularly now with its emphasis on obtaining solutions to real problems—the book should be useful to system analysts, operations researchers, numerical analysts, management scientists, and other specialists from the host of disciplines from which practical optimization applications are drawn. The prerequisites for convenient use of the book are relatively modest; the prime requirement being some familiarity with introductory elements of linear algebra. Certain sections and developments do assume some knowledge of more advanced concepts of linear algebra, such as eigenvector analysis, or some background in sets of real numbers, but the text is structured so that the mainstream of the development can be faithfully pursued without reliance on this more advanced background material.

Although the book is introductory in its level and scope of presentation, it is simultaneously intended to be modern in its approach, reflecting the most recent trends in the foundations of the field. This point is of particular significance since in the last few years these foundations have shifted quite dramatically. A good deal of the modern work in the field, therefore, is concerned not with development of advanced intricate techniques but with development, at an introductory level, of the basic underpinnings of the field. This new work signals the emergence of the field from a collection of isolated theoretical results and heuristic tools into a solidly based discipline in which theory guides the development of techniques. In the past, most of the theory of optimization concentrated on the subject of optimality conditions, and the practical methods of computation were attached to the theoretical body as a somewhat nontheoretical appendage. In the modern setting,

theory and practice are better integrated and we find enhanced interplay between them.

A major connection between theory and practice is convergence analysis—the analysis of how particular iterative solution techniques generate points that converge to a solution of a given problem. In the past, convergence analysis, like the appendage of computational methods to which it applied, could at best be viewed as a series of formulae, which had more relation to particular solution techniques than to problem definition and had little coherence or structure. It is somewhat surprising now to find that convergence theory is not only much more coherent than originally suspected, but that it also helps provide a unifying framework for the field. In essence, we have learned that studying the common properties of various solution techniques for a problem is one of the most effective ways to illuminate the fundamental properties of the problem itself.

The material in this book is organized into three separate parts. Part I is a self-contained introduction to linear programming, a key component of optimization theory. The presentation in this part is fairly conventional, covering the main core of both the underlying theory of linear programming and many of the most effective numerical algorithms. It does not, however, cover those specialized areas of linear programming such as network flows or transportation theory that rely on special structural characteristics. Part II, which is independent of Part I, covers the theory of unconstrained optimization, including both derivations of the appropriate optimality conditions and an introduction to basic algorithms. This part of the book explores the general properties of algorithms and defines various notions of convergence. Part III extends the concepts developed in the second part to constrained optimization problems. Except for a few isolated sections, this part is also independent of Part I. It is possible to go directly into Parts II and III omitting Part I, and, in fact, the book has been used in this way at Stanford on several occasions. Each part of the book contains enough material to form the basis of a one-quarter course. In either classroom use or for self-study, it is important not to overlook the suggested exercises at the end of each chapter. The selections generally include exercises of a computational variety designed to test one's understanding of a particular algorithm, a theoretical variety designed to test one's understanding of a given theoretical development, or of the variety that extends the presentation of the chapter to new areas. One should attempt at least four or five exercises from each chapter. In progressing through the book it would be unusual to read straight through from cover to cover. Generally, one will wish to skip around. In order to facilitate this mode, I have indicated sections of a specialized or digressive nature with an asterisk*.

Development of this book would not have been possible without the help of several individuals and institutions. My perception of linear pro-

gramming was greatly enhanced while working with Adam Shefi, some of whose work I have incorporated into Chapter 5. During the years of writing this book I worked closely with Shmuel Oren, Daniel Gabay, Tsuguhiko Tanahashi, and Verne Chant each of whose many suggestions and comments are an integral part of the final product. I wish to thank Harvey Greenberg, Arthur Geoffrion, and Dimitri Bertsekas, who each made several valuable suggestions which were incorporated into the final manuscript. I wish to acknowledge the Department of Engineering-Economic Systems at Stanford University for supplying a stimulating atmosphere and much of the financial assistance. The effort was also partially supported by the National Science Foundation. I wish to express special thanks to Elaine Christensen who unfledgingly typed several drafts of the manuscript. Finally, as with most textbooks of this type, much is owed to the many students who patiently put up with the difficulties associated with an evolving manuscript and unselfishly contributed to its successful completion.

Washington, D.C. D.G.L.
August 1972

CONTENTS

Chapter 1 INTRODUCTION

1.1 OPTIMIZATION

The concept of optimization is now well-rooted as a principle underlying the analysis of many complex decision or allocation problems. It offers a certain degree of philosophical elegance that is hard to dispute, and it often offers an indispensable degree of operational simplicity. Using this optimization philosophy, one approaches a complex decision problem, involving the selection of values for a number of interrelated variables, by focussing attention on a single objective designed to quantify performance and measure the quality of the decision. This one objective is maximized (or minimized, depending on the formulation) subject to the constraints that may limit the selection of decision variable values. If a suitable single aspect of a problem can be isolated and characterized by an objective, be it profit or loss in a business setting, speed or distance in a physical problem, expected return in the environment of risky investments, or social welfare in the context of government planning, optimization may provide a suitable framework for analysis.

It is, of course, a rare situation in which it is possible to fully represent all the complexities of variable interactions, constraints, and appropriate objectives when faced with a complex decision problem. Thus, as with all quantitative techniques of analysis, a particular optimization formulation should only be regarded as an approximation. Skill in modelling, to capture the essential elements of a problem, and good judgment in the interpretation of results are required to obtain meaningful conclusions. Optimization, then, should be regarded as a tool of conceptualization and analysis rather than as a principle yielding the philosophically correct solution.

Skill and good judgment, with respect to problem formulation and interpretation of results, is enhanced through concrete practical experience and a thorough understanding of relevant theory. Problem formulation

itself always involves a tradeoff between the conflicting objectives of building a mathematical model sufficiently complex to accurately capture the problem description and building a model that is tractable. The expert model builder is facile with both aspects of this tradeoff. One aspiring to become such an expert must learn to identify and capture the important issues of a problem mainly through example and experience; he must learn to distinguish tractable models from nontractable ones through a study of available technique and theory and by nurturing the capability to extend existing theory to new situations.

This book is centered around a certain optimization structure—that characteristic of linear and nonlinear programming. Examples of situations leading to this structure are sprinkled throughout the book, and these examples should help to indicate how practical problems can be often fruitfully structured in this form. The book mainly, however, is concerned with the development, analysis, and comparison of algorithms for solving general sub-classes of optimization problems. This is valuable not only for the algorithms themselves, which enable one to solve given problems, but also because identification of the collection of structures they most effectively solve can enhance one's ability to formulate problems.

1.2 TYPES OF PROBLEMS

The content of this book is broken down into three major parts: Linear Programming, Unconstrained Problems, and Constrained Problems. The last two parts together comprise the subject of nonlinear programming.

Linear Programming

Linear programming is without doubt the most natural mechanism for formulating a vast array of problems with modest effort. A linear programming problem is characterized, as the name implies, by linear functions of the unknowns; the objective is linear in the unknowns, and the constraints are linear equalities or linear inequalities in the unknowns. One familiar with other branches of linear mathematics might suspect, initially, that linear programming formulations are popular because the mathematics is nicer, the theory is richer, and the computation simpler for linear problems than for nonlinear ones. But, in fact, these are *not* the primary reasons. In terms of mathematical and computational properties, there are much broader classes of optimization problems than linear programming problems that have elegant and potent theories and for which effective algorithms are available. It seems that the popularity of linear programming lies primarily with the formulation phase of analysis rather than the solution phase—and for good cause. For one thing, a great number of constraints and objectives that arise in practice *are* indisputably linear.

Thus, for example, if one formulates a problem with a budget constraint restricting the total amount of money to be allocated among two different commodities, the budget constraint takes the form $x_1 + x_2 \leqslant B$, where x_i, $i = 1, 2$, is the amount allocated to activity i, and B is the budget. Similarly, if the objective is, for example, maximum weight, then it can be expressed as $w_1x_1 + w_2x_2$, where w_i, $i = 1, 2$, is the unit weight of the commodity i. The overall problem would be expressed as

$$\begin{aligned} \text{maximize} \quad & w_1x_1 + w_2x_2 \\ \text{subject to} \quad & x_1 + x_2 \leqslant B, \\ & x_1 \geqslant 0, \quad x_2 \geqslant 0 \end{aligned}$$

which is an elementary linear program. The linearity of the budget constraint is extremely natural in this case and does not represent simply an approximation to a more general functional form.

Another reason that linear forms for constraints and objectives are so popular in problem formulation is that they are often the least difficult to define. Thus, even if an objective function is not purely linear by virtue of its inherent definition (as in the above example), it is often far easier to define it as being linear than to decide on some other functional form and convince others that the more complex form is the best possible choice. Linearity, therefore, by virtue of its simplicity, often is selected as the easy way out or, when seeking generality, as the only functional form that will be equally applicable (or nonapplicable) in a class of similar problems.

Of course, the theoretical and computational aspects *do* take on a somewhat special character for linear programming problems—the most significant development being the simplex method. This algorithm is developed in Chapters 2 and 3 and occupies most of the attention that we devote to linear programming.

Unconstrained Problems

It may seem that unconstrained optimization problems are so devoid of structural properties as to preclude their applicability as useful models of meaningful problems. Quite the contrary is true for two reasons. First, it can be argued, quite convincingly, that if the scope of a problem is broadened to the consideration of all relevant decision variables, there may then be no constraints—or put another way, constraints represent artificial delimitations of scope, and when the scope is broadened the constraints vanish. Thus, for example, it may be argued that a budget constraint is not characteristic of a meaningful problem formulation; since by borrowing at some interest rate it is always possible to obtain additional funds, and hence rather than introducing a budget constraint, a term reflecting the cost of funds should be incorporated into the objective. A similar argument

applies to constraints describing the availability of other resources which at some cost (however great) could be supplemented.

The second reason that many important problems can be regarded as having no constraints is that constrained problems are sometimes easily converted to unconstrained problems. For instance, the sole effect of equality constraints is simply to limit the degrees of freedom, by essentially making some variables functions of others. These dependencies can sometimes be explicitly characterized, and a new problem having its number of variables equal to the true degree of freedom can be determined. As a simple specific example, a constraint of the form $x_1 + x_2 = B$ can be eliminated by substituting $x_2 = B - x_1$ everywhere else that x_2 appears in the problem.

Aside from representing a significant class of practical problems, the study of unconstrained problems, of course, provides a stepping stone toward the more general case of constrained problems. Many aspects of both theory and algorithms are most naturally motivated and verified for the unconstrained case before progressing to the constrained case.

Constrained Problems

In spite of the arguments given above, many problems met in practice are formulated as constrained problems. This is because in most instances a complex problem such as, for example, the detailed production policy of a giant corporation, the planning of a large government agency, or even the design of a complex device cannot be directly treated in its entirety accounting for all possible choices, but instead must be decomposed into separate subproblems—each subproblem having constraints which are imposed to restrict its scope. Thus, in planning problems, budget constraints are commonly imposed in order to decouple that one problem from a more global one. Therefore, one frequently encounters general nonlinear constrained mathematical programming problems.

The general mathematical programming problem can be stated as

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) \\ &\text{subject to} && h_i(\mathbf{x}) = 0, \qquad i = 1, 2, \ldots, m \\ &&& g_j(\mathbf{x}) \leqslant 0, \qquad j = 1, 2, \ldots, r \\ &&& \mathbf{x} \in S. \end{aligned}$$

In this formulation, $\mathbf{x}$ is an n-dimensional vector of unknowns, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, and f, h_i, $i = 1, 2, \ldots, m$, and g_j, $j = 1, 2, \ldots, r$ are real-valued functions of the variables $x_1, x_2, \ldots, x_n$. The set S is a subset of n-dimensional space. The function f is the *objective function* of the problem and the equations, inequalities, and set restrictions are *constraints*.

Generally, in this book, additional assumptions are introduced in order to make the problem smooth in some suitable sense. For example, the functions in the problem are usually required to be continuous, or perhaps to have continuous derivatives. This ensures that small changes in $\mathbf{x}$ lead to small changes in other values associated with the problem. Also, the set S is not allowed to be arbitrary but usually is required to be a connected region of n-dimensional space, rather than, for example, a set of distinct isolated points. This ensures that small changes in $\mathbf{x}$ can be made. Indeed, in a majority of problems treated, the set S is taken to be the entire space; there is no set restriction.

In view of these smoothness assumptions, one might characterize the problems treated in this book as *continuous variable programming*, since we generally discuss problems where all variables and function values can be varied continuously. In fact, this assumption forms the basis of many of the algorithms discussed, which operate essentially by making a series of small movements in the unknown $\mathbf{x}$ vector.

1.3 SIZE OF PROBLEMS

One obvious measure of the complexity of a programming problem is its size, measured in terms of the number of unknown variables or the number of constraints. As might be expected, the size of problems that can be effectively solved has been increasing with advancing computing technology and with advancing theory. Today, with present computing capabilities, however, it is reasonable to distinguish three classes of problems: *small-scale problems* having about five or less unknowns and constraints; *intermediate-scale problems* having from about five to a hundred variables; and *large-scale problems* having on the order of a thousand variables and constraints. This classification is not entirely rigid, but it reflects at least roughly not only size but the basic differences in approach that accompany different size problems.

Much of the early theory associated with optimization, particularly in nonlinear programming, is directed at obtaining necessary and sufficient conditions satisfied by a solution point, rather than at questions of computation. This theory involves mainly the study of Lagrange multipliers, including the Kuhn–Tucker Theorem and its extensions. It tremendously enhances insight into the philosophy of constrained optimization and provides satisfactory basic foundations for other important disciplines, such as the theory of the firm, consumer economics, and optimal control theory. The interpretation of Lagrange multipliers that accompanies this theory is valuable in virtually every optimization setting. As a theoretical basis for computing numerical solutions to optimization, however, this early theory is applicable only to *small-scale*

programming problems, because the equations resulting from the necessary conditions can be efficiently solved directly only when the problem is small enough for hand calculation.

If it is acknowledged from the outset that a given problem is too large and too complex to be efficiently solved by hand (and hence it is acknowledged that a computer solution is desirable), then one's theory should be directed toward development of procedures that exploit the efficiencies of computers. In most cases this leads to the abandonment of the idea of solving the set of necessary conditions in favor of the more direct procedure of searching through the space (in an intelligent manner) for ever-improving points. For the development and implementation of such search procedures, the theory of necessary conditions is of little value in itself, but as repeatedly demonstrated throughout this book, it forms the core of a more complete theory that is applicable to such procedures.

Today, search techniques can be effectively applied to more or less general nonlinear programming problems having on the order of 100 to 200 variables, and to linear programming problems having about 400 constraints and 1000 variables. This range of problem that can be solved by a computer with a search method we refer to as *intermediate-scale programming*.

Problems of even greater size, *large-scale programming* problems, can be solved if they possess special structural characteristics that can be exploited by a solution method. The study of large-scale programming consists of the identification of important special structures and the development of techniques that exploit these structures. It is, therefore, a more problem-dependent body of theory than the other theoretical aspects of programming problems.

This book is devoted mainly to the theory of intermediate-scale programming and therefore can be considered as focusing on the aspect of general theory that is most fruitful for computation in the widest class of problems. While necessary and sufficient conditions are examined and their application to small-scale problems is illustrated, our primary interest in such conditions is in their role as the core of a broader theory applicable to the solution of larger problems. At the other extreme, although some instances of structure exploitation are discussed, we focus primarily on the general continuous variable programming problem rather than on special techniques for special structures.

1.4 ITERATIVE ALGORITHMS AND CONVERGENCE

The most important characteristic of a high-speed digital computer is its ability to perform repetitive operations efficiently, and in order to exploit this basic characteristic, most algorithms designed to solve large

optimization problems are iterative in nature. Typically, in seeking a vector that solves the programming problem, an initial vector $\mathbf{x}_0$ is selected and the algorithm generates an improved vector $\mathbf{x}_1$. The process is repeated and a still better solution $\mathbf{x}_2$ is found. Continuing in this fashion, a sequence of ever-improving points $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k, \ldots$, is found that approaches a solution point $\mathbf{x}^*$. For linear programming problems, the generated sequence is of finite length, reaching the solution point exactly after a finite (although initially unspecified) number of steps. For nonlinear programming problems, the sequence generally does not ever exactly reach the solution point, but converges toward it. In operation, for nonlinear problems, the process is terminated when a point sufficiently close to the solution point, for practical purposes, is obtained.

The theory of iterative algorithms can be divided into three (somewhat overlapping) aspects. The first is concerned with the creation of the algorithms themselves. Algorithms are not conceived arbitrarily, but are based on a creative examination of the programming problem, its inherent structure, and the efficiencies of digital computers. The second aspect is the verification that a given algorithm will in fact generate a sequence that converges to a solution point. This aspect is referred to as *global convergence analysis*, since it addresses the important question of whether the algorithm, when initiated far from the solution point, will eventually converge to it. The third aspect is referred to as *local convergence analysis* and is concerned with the rate at which the generated sequence of points converges to the solution. One cannot regard a problem as solved simply because an algorithm is known which will converge to the solution, since it may require an exorbitant amount of time to reduce the error to an acceptable tolerance. It is essential when prescribing algorithms that some estimate of the time required be available. It is the convergence-rate aspect of the theory that allows some quantitative evaluation and comparison of different algorithms, and at least crudely, assigns a measure of tractability to a problem, as discussed in Section 1.1.

A modern-day technical version of Confucius' most famous saying, and one which represents an underlying philosophy of this book, might be, "One good theory is worth a thousand computer runs". Thus the convergence properties of an iterative algorithm can be estimated with confidence either by performing numerous computer experiments on different problems or by a simple well-directed theoretical analysis. A simple theory, of course, provides invaluable insight as well as the desired estimate.

It is perhaps somewhat surprising that there does not yet exist a useful convergence theory for the simplex method of linear programming, one of the oldest and most important optimization techniques. This seems to be due to the fact that, since convergence occurs in a finite number of steps, an estimate of the total number of steps, rather than a rate of convergence,