DRAFT
Formulation and Analysis of Linear Programs

Benjamin Van Roy and Kahn Mason

September 26, 2005

Contents

Chapter 1

Introduction

Optimization is the process of selecting the best among a set of alternatives. An optimization problem is characterized by a set of feasible solutions and an objective function, which assigns a measure of utility to each feasible solution. A simple approach to optimization involves listing the feasible solutions, applying the objective function to each, and choosing one that attains the optimum, which could be the maximum or minimum, depending on what we are looking for. However, this approach does not work for most interesting problems. The reason is that there are usually too many feasible solutions, as we illustrate with the following example.

Example 1.0.1. (Task Assignment) *Suppose we are managing a group of* 20 *employees and have* 20 *tasks that we would like to complete. Each employee is capable of completing only a subset of the tasks. Furthermore, each employee can only be assigned one task. We need to decide on which task to assign to each employee. The set of feasible solutions here is the set of* 20! *possible assignments of the* 20 *tasks to the* 20 *employees.*

Suppose our objective is to maximize the number of tasks accomplished. One way of doing this is to enumerate the 20! *possible assignments, calculate the number of tasks that would be completed in each case, and choose an assignment that maximizes this quantity. However,* 20! *is a huge number – its greater than a billion billion. Even if we used a computer that could assess* 1 *billion assignments per second, iterating through all of them would take more than* 75 *years!*

Fortunately, solving an optimization problem does not always require iterating through and assessing every feasible solution. This book is about linear programs – a class of optimization problems that can be solved very quickly by numerical algorithms. In a linear program, solutions are encoded in terms of real-valued decision variables. The objective function is linear in the decision variables, and the feasible set is constrained by linear inequalities. Many important practical problems can be formulated as linear programs. For example, the task-assignment problem of Example 1.0.1 can be formulated as a linear program and solved in a small fraction of a second by a linear programming algorithm running on just about any modern personal computer.

This book represents a first course on linear programs, intended to serve engineers, economists, managers, and other professionals who think about design, resource allocation, strategy, logistics, or other decisions. The aim is to develop two intellectual assets. The first is an ability to formulate a wide variety of practical problems as linear programs. The second is a geometric intuition for linear programs. The latter helps in interpreting solutions to linear programs and often also leads to useful insights about problems formulated as linear programs. Many other books focus on linear programming algorithms. We discuss such algorithms in our final chapter, but they do not represent the focus of this book. The following sections offer previews of what is to come in subsequent chapters.

1.1 Linear Algebra

Linear algebra is about systems of linear equations. It forms a foundation for linear programming, which deals with both linear equations and inequalities. Chapter 2 presents a few concepts from linear algebra that are essential to developments in the remainder of the book. The chapter certainly does not provide comprehensive coverage of important topics in linear algebra; rather, the emphasis is on two geometric interpretations of systems of linear equations.

The first associates each linear equation in n unknowns with an $(n-1)$-dimensional plane consisting of points that satisfy the equation. The set of solutions to a system of m linear equations in n unknowns is then the intersection of m planes. A second perspective views solving a system of linear equations in terms of finding a linear combination of n vectors that generates another desired vector.

Though these geometric interpretations of linear systems of equations are simple and intuitive, they are remarkably useful for developing insights about real problems. In Chapter 2, we will illustrate the utility of these concepts through examples involving contingent claims analysis and production. In contingent claims analysis, the few ideas we discuss about linear algebra lead to an understanding of asset replication, market completeness, and state prices. The production example sets the stage for our study of linear programming.

1.2 Linear Programs

In Chapter 3, we introduce linear programs. In a linear program, a solution is defined by n real numbers called *decision variables*. The set of feasible solutions is a subset of the n-dimensional space, characterized by linear inequalities. Each linear inequality is satisfied by points in the n-dimensional space that are on one side of an $(n-1)$-dimensional plane, forming a set called a *half space*. Given m inequalities, the set of feasible solutions is the intersection of m half-spaces. Such a set is called a *polyhedron*. A linear program involves optimization (maximization or minimization) of a linear function of the decision variables over the polyhedron of feasible solutions. Let us bring this to life with an example.

Example 1.2.1. (Petroleum Production) *Crude petroleum extracted from a well contains a complex mixture of component hydrocarbons, each with a different boiling point. A refinery separates these component hydrocarbons using a distillation column. The resulting components are then used to manufacture consumer products such as low, medium, and high octane gasoline, diesel fuel, aviation fuel, and heating oil.*

Suppose we are managing a company that manufactures n petroleum products and have to decide on the number of liters x_j, $j = 1, \ldots, n$ of each product to manufacture next month. Since the amount we produce must be positive, we constrain the decision variables by linear inequalities $x_j \geq 0$, $j = 1, \ldots, n$. Additional constraints arise from the fact that we have limited resources. We have m types of resources in the form of component hydrocarbons. Let b_i, $i = 1, \ldots, m$, denote the quantity in liters of the ith resource to be available to us next month. Our process for manufacturing the jth product consumes a_{ij} liters of the ith resource. Hence, production of quantities $x_1, \ldots, x_n$, requires $\sum_{j=1}^n a_{ij} x_j$ liters of resource i. Because we only have b_i liters of resource i available, our solution must satisfy

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i.$$

We define as our objective maximization of next month's profit. Given that the jth product garners c_j Dollars per liter, production of quantities $x_1, \ldots, x_n$ would generate $\sum_{j=1}^n c_j x_j$ Dollars in profit. Assembling constraints *and the objective, we have the following linear program:*

$$\begin{array}{lll} \text{maximize} & \sum_{j=1}^{n} c_j x_j & \\ \text{subject to} & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, & i = 1, \ldots, m \\ & x_j \geq 0 & j = 1, \ldots, n. \end{array}$$

The above example illustrates the process of formulating a linear program. Linear inequalities characterize the polyhedron of feasible solutions and a linear function defines our objective. Resource constraints limit the objective.

1.3 Duality

Associated with every linear program is another called the *dual*. In order to distinguish a linear program from its dual, the former is called the *primal*. As an example of a *dual* linear program, we describe that associated with the petroleum production problem, for which the primal was formulated in Example 1.2.1.

Example 1.3.1. (Dual of Petroleum Production) *The dual linear program comes from a thought experiment about prices at which we would be willing to sell all our resources. Consider a set of resource prices* $y_1, \ldots, y_m$, *expressed in terms of Dollars per liter. To sell a resource, we would clearly require that the price is nonnegative:* $y_i \geq 0$. *Furthermore, to sell a bundle of resources that could be used to manufacture a liter of product* j *we would require that the proceeds* $\sum_{i=1}^{m} y_i a_{ij}$ *at least equal the revenue* c_j *that we would receive for the finished product. Aggregating constraints, we have a characterization of the set of prices at which we would be willing to sell our entire stock of resources rather than manufacture products:*

$$\begin{array}{ll} \sum_{i=1}^{m} y_i a_{ij} \geq c_j, & j = 1, \ldots, n \\ y_i \geq 0 & i = 1, \ldots, m. \end{array}$$

The dual linear program finds our minimum net proceeds among such situations:

$$\begin{array}{lll} \text{minimize} & \sum_{i=1}^{m} y_i b_i & \\ \text{subject to} & \sum_{i=1}^{m} y_i a_{ij} \geq c_j, & j = 1, \ldots, n \\ & y_i \geq 0 & i = 1, \ldots, m. \end{array}$$

Duality theory studies the relationship between primal and dual linear programs. A central result – the duality theorem – states that optimal objective values from the primal and dual are equal, if both are finite. In our petroleum production problem, this means that, in the worst case among situations that would induce us to sell all our resources rather than manufacture, our proceeds would equal the revenue we could earn from manufacturing.

In Chapter 4, we develop duality theory. This theory extends geometric concepts of linear algebra, and like linear algebra, duality leads to useful insights about many problems. For example, duality is central to understanding situations where adversaries compete, as we now discuss.

1.4 Two-Player Zero-Sum Games

A two player zero-sum, or competitive, game considers the situation where two agents are dividing a good. Examples include two companies competing over a market of constant size, or games of chess, football, or rock-paper-scissors.

In many situations the game can be described as follows. Each agent picks an action. If agent 1 picks action i and agent 2 picks action j then agent 1 receives r_{ij} units of benefit and agent 2 receives $-r_{ij}$ units of benefit. The game is said to be *zero-sum* because the benefits received by the two players always sum to zero, since one player receives r_{ij} and the other $-r_{ij}$.

Agent 1 wants to maximize r_{ij}, while agent 2 wants to maximize $-r_{ij}$ (or, equivalently, minimize r_{ij}). What makes games like this tricky is that agent 1 must take into account the likely actions of agent 2, who must in turn take into account the likely actions of agent 1. For example, consider a game of rock-paper-scissors, and suppose that agent 1 always chooses rock. Agent 2 would quickly learn this, and choose a strategy that will always beat agent 1, namely, pick paper. Agent 1 will then take this into account, and decide to always pick scissors, and so on.

The circularity that appears in our discussion of rock-paper-scissors is associated with the use of *deterministic strategies* – strategies that selects a single action. If the opposing player can learn an agent's strategy, he can design a strategy that will beat it. One way around this problem is to consider randomized strategies. For example, agent 1 could randomly choose between rock, paper, and scissors each time. This would prevent agent 2 from predicting the action of agent 1, and therefore, from designing an effective counter strategy.

With randomized strategies, the payoff will still be r_{ij} to agent 1, and $-r_{ij}$ to player 2, but now i and j are randomly (and independently) chosen. Agent 1 now tries to maximize the expected value of r_{ij}, while player 2 tries to maximize the expected value of $-r_{ij}$.

Example 1.4.1. (Drug running) *A South American drug lord is trying to* *get as many of his shipments across the border as possible. He has a fleet of boats available to him, and each time he sends a boat, he can choose one of three ports at which to unload. He could choose to unload in Miami, Los Angeles, or San Francisco.*

The USA Coastguard is trying to intercept as many of the drug shipments as possible but only have sufficient resources to cover one port at a time. Moreover, the chance of intercepting a drug shipment differs from port to port. A boat arriving at a port closer to South America will have more fuel with which to evade capture than one arriving further away. The probabilities of interception are given by the following table.

Port	Probability of interception
Miami	*1/3*
Los Angeles	*1/2*
San Francisco	*3/4*

The drug lord considers sending each boat to Miami, but the coastguard realizing this would always choose to cover Miami, and 2/3 *of his boats would get through. A better strategy would be to pick a port at random (each one picked with* 1/3 *probability). Then, the coastguard should cover port 3, since this would maximize the number of shipments captured. In this scenario,* 3/4 *of the shipments would get through, which is better than* 2/3*. But is this the best strategy?*

If we allow for randomized strategies, the problem of finding a strategy that maximizes an agents expected payoff against the best possible counter-strategy can be formulated as a linear program. Interesting, the formulation from the perspective of the first agent is the dual of the formulation from the perspective of the second agent. We will see that duality guarantees that there are a pair of strategies for agents 1 and 2 that put them in equilibrium. In particular, when these strategies are employed, agent 1 can not improve her strategy benefit by learning the strategy of agent 2, and vice-versa. This is also treated in Chapter 4 as a special case of duality.

1.5 Network Flows

Network flow problems are a class of optimization problems that arise in many application domains, including the analysis of communication, transportation, and logistic networks. Let us provide an example.

Example 1.5.1. (Transportation Problem) *Suppose we are selling a single product and have inventories of* $s_1, \ldots, s_m$ *in stock at* m *different ware-*

house locations. We plan to ship our entire inventory to n customers, in quantities of $d_1, \ldots, d_n$. Shipping from location i to customer j costs c_{ij} Dollars. We need to decide on how to route shipments.

Our decision variables x_{ij} represent the number of units to be shipped from location i to customer j. The set of feasible solutions is characterized by three sets of constraints. First, the amount shipped from each location to each customer is not negative: $x_{ij} \geq 0$. Second, all inventory at location i is shipped: $\sum_{j=1}^{n} x_{ij} = s_i$. Finally, the number of units sold to customer j are shipped to him: $\sum_{i=1}^{m} x_{ij} = d_j$. Note that the last two constraints imply that the quantity shipped is equal to that received: $\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j$.

With an objective of minimizing total shipping costs, we arrive at the following linear program:

$$\begin{array}{lll}
\text{minimize} & \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} & \\
\text{subject to} & \sum_{j=1}^{n} x_{ij} = s_i, & i = 1, \ldots, m \\
& \sum_{i=1}^{m} x_{ij} = d_j, & j = 1, \ldots, n \\
& x_{ij} \geq 0 & i = 1, \ldots, m; j = 1, \ldots, n.
\end{array}$$

The set of feasible solutions to the transportation problem allows for shipment of fractional quantities. This seems problematic if units of the product are indivisible. We don't want to send a customer two halves of a car from two different warehouses! Surprisingly, we never run into this problem because shipping costs in the linear program are guaranteed to be minimized by integral shipments when inventories $s_1, \ldots, s_m$ and customer sales $d_1, \ldots, d_n$ are integers.

The situation generalizes to network flow problems of which the transportation problem is a special case – when certain problem parameters are integers, the associated linear program is optimized by an integral solution. This enables use of linear programming – a formulation involving continuous decision variables – to solve a variety of optimization problems with discrete decision variables. An example is the task-assignment problem of Example 1.0.1. Though the problem is inherently discrete – either we assign a task to an employee or we don't – it can be solved efficiently via linear programming. In Chapter 5, we discuss a variety of network flow problems and the integrality of optimal solutions.

1.6 Markov Decision Problems

Markov decision problems involve systems that evolve in time. In each time period, the system can be in one of a number of states, and based on the state, a decision is made that influences the future evolution of the system. The state of the system in the next time period depends only upon the current state and decision.

Markov decision problems arise in a variety of applications, from epidemiology to queueing to the design of interactive robots.

Example 1.6.1. (Machine replacement) *Suppose we wish to minimize the average cost of running a machine over time. At each time period, the machine can be in one of states* $1, \ldots, n$ *where state* 1 *represents a a perfect condition, and higher-valued states represent greater degrees of deterioration.*

The cost of running a machine in state i *for one time period is* $c(i)$*. The cost is increasing in* i *– it costs more to run a deteriorated machine. As a machine is used, it deteriorates at a random rate. In each time step, if the state* i *of the machine is less than* n*, there is a probability* p *that the state increases to* $i + 1$*. Once in state* n*, the machine remains in this highly deteriorated state.*

At each time period, based on the state of the machine, we can decide to replace it. Replacing the machine incurs a cost of C *and leaves us with a machine in state* 1 *in the next time period. We wish to design a decision strategy that minimizes our time-averaged cost.*

We discuss in Chapter **??** how Markov decision problems like the machine replacement problem can be solved using linear programming.

1.7 Linear Programming Algorithms

As we have discussed, many important problems can be formulated as linear programs. Another piece of good news is that there are algorithms that can solve linear programs very quickly, even with thousands and sometimes millions of variables and constraints. The history of linear programming algorithms is a long winding road marked with fabulous ideas. Amazingly, over the course of their development over the past five decades, linear programming algorithms have become about a thousand times faster and state-of-the-art computers have become about a thousand times faster – together this makes a factor of about one million. A linear program that required one year to solve when the first linear programming algorithms were developed now takes about half a minute! In Chapter **??**, we discuss some of this history and present some of the main algorithmic ideas in linear programming.