CHAPTER 1: LINEAR PROGRAMMING

Overview

Civil and Environmental (CEE) Systems Analysis refers to the development and solution of optimization problems that guide decision making and planning in CEE applications. In this course, you will learn to abstract mathematical optimization programs from physical systems to "optimally" design a civil engineered system.

Our exposition begins with an introduction of the canonical "optimization program", or just "optimization problem". This mathematical statement formally communicates the optimization problem, and is a structure we use throughout the course. We discuss its essential elements, namely the optimization objective function and constraints, and their physical significance. The remainder of this chapter covers the simplest of optimization problems - linear programming (LP). Put simply, the class of LPs includes all optimization problems in which the objective and constraint functions are linear. Coincidentally, a large number of CEE system design problems can be formulated and solved as a LP. We discuss several examples, and provide both graphical and algorithmic solution examples. During our exposition, several concepts fundamental to all optimization problems are introduced, including feasible sets, boundedness, uniqueness, constraint domination, and active constraints.

Readers should note that optimization is a fundamental discipline that is highly applicable to a variety of fields, including CEE, mechanical engineering, industrial engineering, electrical engineering, computer science, materials science engineering, bioengineering, aerospace engineering, chemical engineering, economics, physics, and so forth. It is an essential tool for the modern engineer, made more practical by the proliferation of numerical computation. By the course's conclusion, you will be capable of abstracting nearly any application design problem into a mathematical optimization problem, solve it, and perform a variety of analyses.

Chapter Organization

This chapter is organized as follows:

- (Section 1) Objective Functions and constraints
- (Section 2) Linear Programs (LP)
- (Section 3) Example 1: Transportation Problem
- (Section 4) Example 2: Shortest Path Problem
- (Section 5) Graphical Solutions to LPs
- (Section 6) Simplex Algorithm

1 Objective Function and Constraints

We begin our exposition of optimization by introducing the canonical optimization problem. The canonical problem is to minimize a cost (or objective) function

$$\min_x f(x), \tag{1}$$

subject to constraints:

$$g_i(x) \leq 0, \quad i = 1, \cdots, m \tag{2}$$
$$h_j(x) = 0, \quad j = 1, \cdots, l, \tag{3}$$

where (2) includes the *inequality constraints* and (3) includes the *equality constraints*. The exact structure of x, $f(x)$, $g_i(x)$, and $h_j(x)$ determine the 'type' of optimization program, such as linear program, quadratic program, integer program, and nonlinear program. As such (1)-(3) essentially summarizes the entire course. Next we examine the notation and physical interpretation of objective functions and constraints.

1.1 Objective Functions

In (1), variable x represents the *decision variable*. This is a vector of system parameters that we seek to design to meet some objective. For example, vector x may include a beam width, concrete density, or rod radius. It might include chemical concentration, water volume, or fluid pressure. Mathematically, we say $x \in \mathbb{R}^n$. This means x exists within the domain of n-element vectors whose elements are real numbers. In applications, we may restrict x to a subset of real numbers $\mathcal{D}$, i.e. $x \in \mathcal{D} \subseteq \mathbb{R}^n$. For example, the beam width, concrete density, and rod radius can only be positive numbers.

The function $f(x)$ maps the decision variables x into a performance metric. Mathematically, $f(x) : \mathbb{R}^n \to \mathbb{R}$. This is, the objective function measures the performance of your decision variables using some scalar metric. Example objectives include monetary cost [USD], fuel consumption [gal], emissions [g], and power [kW].

Remark 1.1 (Objective Function Terminology)**.** Note that "cost" function, "reward" function, and "objective" function are used interchangeably. The objective function need not represent an economic quantity. However, there are some conventions. The term "cost" has the connotation of minimizing an objective function, whereas the term "reward" has the connotation of maximizing an objective function.

Remark 1.2 (Maximizing a Reward)**.** Note that maximizing any "reward" function f(x) can be for-

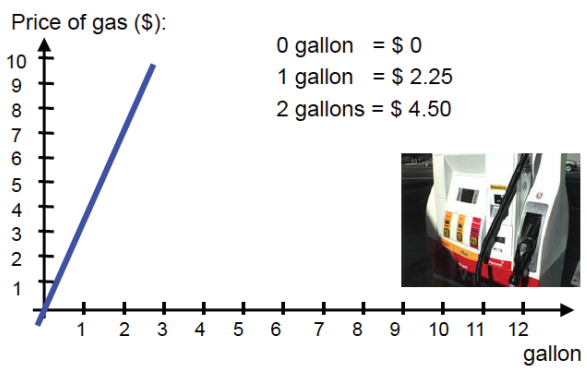

Figure 1: Gasoline has linear cost.

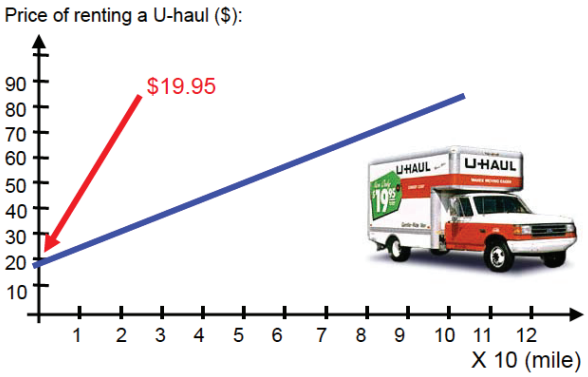

Figure 2: U-haul rental has affine cost.

mulated as a minimization problem by minimizing it's negative. That is,

$$x^* = \arg\max_x f(x) = \arg\min_x -f(x) \tag{4}$$

where x^* represents the optimal design.

The mathematical structure of objective function $f(x)$ has relevance throughout the course. At this point, let us examine linear and affine objective functions. Consider the cost of gasoline versus the cost of renting a U-haul, as depicted in Fig. 1 and 2. The price of gasoline goes through the origin, i.e. zero gallons cost zero dollars. A U-haul rental includes a one-time price plus a price per mile driven. Mathematically, these can be written:

$$\text{Linear Function:} \quad f(x) = ax, \tag{5}$$

$$\text{Affine Function:} \quad f(x) = bx + c. \tag{6}$$

Proposition 1 (Linear vs. Affine Cost Functions)**.** Next we conjecture that minimizing an affine cost function is equivalent to minimizing the linear part only. That is,

$$x^* = \arg\min_x \{bx + c\} = \arg\min_x \{bx\}\,. \tag{7}$$

More generally, minimizing any nonlinear cost function is equivalent to minimizing the same cost function plus an offset term. That is,

$$x^* = \arg\min_x \{f(x)\} = \arg\min_x \{f(x) + C\}\,. \tag{8}$$

As a consequence, we can always disregard the affine offset term when formulating an optimization problem. To understand, consider the graphical explanation in Fig. 3. This example demonstrates that translating the objective function value by a constant does not impact the minimizer

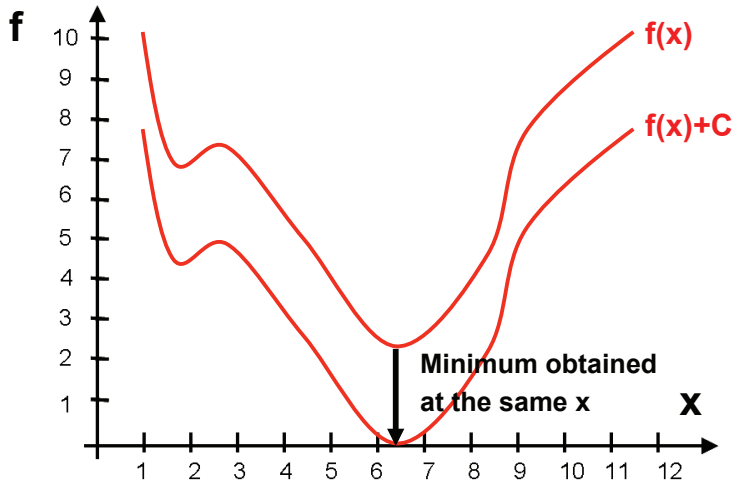

Figure 3: Minimizing any nonlinear cost function is equivalent to minimizing the same cost function plus an offset term.

value. For example, minimizing the affine cost function

$$f(x_1, x_2) = 2x_1 + 3x_2 + 5 \tag{9}$$

is equivalent to minimizing the linear cost function

$$f(x_1, x_2) = 2x_1 + 3x_2 \tag{10}$$

Exercise 1. *Which of the following functions are linear, affine, neither, or both, over the set* $D = [-10, 10]$*?*

(a) $f(x) = 0$

(b) $f(x) = x$

(c) $f(x) = x^2$

(d) $f(x) = -x^2$

(e) $f(x) = x^3$

(f) $f(x) = \sin(x)$

(g) $f(x) = e^{-x^2}$

(h) $f(x) = |x|$

1.2 Constraints

Constraints encode physical restrictions on the decision variables. As such, it is natural to explain the concept of constraints via an example.

Example 1.1 (Building a Wall)**.** Suppose you are tasked with building a wall for minimum cost, subject to certain design specifications. Specifically, you must decide how much cement and steel

beam to use. You are given the problem parameters listed in Table 1.

Table 1: Building a Wall Problem Parameters

Cost of a pound of cement (USD per kg)	a_1
Cost of a foot of steel beam (USD per m)	a_2
Weight of cement (kg)	x_1
Length of steel beam (m)	x_2
Total cost (USD)	$f(x_1, x_2) = a_1 x_1 + a_2 x_2$

Note that all variables have different dimensions. However, the expressions $a_1 x_1$, $a_2 x_2$, and $f(x_1, x_2)$ have the same units - USD. Our goal is to minimize the total cost, subject to certain constraints

- Your maximum budget for cement is $c_{\max}$: $a_1 x_1 \leq c_{\max}$
- Your minimum budget for steel is $s_{\min}$: $a_2 x_2 \geq s_{\min}$
- Your cannot have negative kg of cement, nor negative m of steel beam: $x_1, x_2 \geq 0$
- Your maximum total budget is $f_{\max}$: $a_1 x_1 + a_2 x_2 \leq f_{\max}$

The optimization program incorporating all the constraints can be formulated as:

$$\begin{aligned} \text{Minimize:} \quad & f(x_1, x_2) = a_1 x_1 + a_2 x_2 && (11)\\ \text{Subject to:} \quad & a_1 x_1 \leq c_{\max} && (12)\\ & a_2 x_2 \geq s_{\min} && (13)\\ & x_1, x_2 \geq 0 && (14)\\ & a_1 x_1 + a_2 x_2 \leq f_{\max} && (15) \end{aligned}$$

We also remark that constraints are often represented in *negative null form* or *standard form*, as shown in Table 2. In negative-null form, all inequality constraints are converted into less-than-or-equal-to constraints, with zeros on the right-hand-size. In standard form, all inequality constraints are converted into less-than-or-equal-to constraints, with the constants moved to the right-hand-size.

Often, physical constraints take the mathematical form of equalities. We call these *equality constraints*. For example, consider the additional constraint:

- You must spend exactly twice as much for steel as for cement: $a_2 x_2 = 2 a_1 x_1$

Table 2: Negative-null and Standard Forms

Negative-null form	Standard form
$a_1 x_1 - c_{\max} \leq 0$	$a_1 x_1 \leq c_{\max}$
$s_{\min} - a_2 x_2 \leq 0$	$-a_2 x_2 \leq -s_{\min}$
$-x_1 \leq 0$	$-x_1 \leq 0$
$-x_2 \leq 0$	$-x_2 \leq 0$
$a_1 x_1 + a_2 x_2 - f_{\max} \leq 0$	$a_1 x_1 + a_2 x_2 \leq f_{\max}$

Remark 1.3 (Equality Constraints $\rightarrow$ Inequality Constraints)**.** Note that equality constraint $a_2 x_2 = 2a_1 x_1$ is equivalent to

$$a_2 x_2 \geq 2a_1 x_1 \qquad \text{AND} \qquad a_2 x_2 \leq 2a_1 x_1.$$

Consequently, any equality constraint can be converted into two inequality constraints, without loss of generality or accuracy. As a result, we can express the optimization program as

$$\begin{aligned}
\text{Minimize:} \quad & f(x_1, x_2) = a_1 x_1 + a_2 x_2 && (16)\\
\text{Subject to:} \quad & a_1 x_1 \leq c_{\max} && (17)\\
& a_2 x_2 \geq s_{\min} && (18)\\
& x_1, x_2 \geq 0 && (19)\\
& a_1 x_1 + a_2 x_2 \leq f_{\max} && (20)\\
& a_2 x_2 \geq 2a_1 x_1 && (21)\\
& a_2 x_2 \leq 2a_1 x_1 && (22)
\end{aligned}$$

with inequality constraints only.

Remark 1.4 (Equality Constraints $\rightarrow$ Program Reducton)**.** Interestingly, equality constraints can be used to reduce the program size. For example, we can solve the equality constraint for x_2,

$$x_2 = \frac{2a_1}{a_2} x_1 \tag{23}$$

Substituting (23) into (11)-(15) renders the reduced program

$$\begin{aligned} \text{Minimize:} \quad & f(x_1) = 3a_1x_1 && (24)\\ \text{Subject to:} \quad & a_1x_1 \leq c_{\max} && (25)\\ & 2a_1x_1 \geq s_{\min} && (26)\\ & x_1, x_2 \geq 0 && (27)\\ & 3a_1x_1 \leq f_{\max} && (28) \end{aligned}$$

which contains only one decision variable, x_1. From (24), it is intuitive that we need to minimize x_1 until some lower bound becomes active. Assuming $s_{\min}$ is positive (which must be true to be physically meaningful), then (26) is true with equality at the minimizer. That is, this inequality constraint defines the minimum.

$$x_1^* = \arg\min_{x_1} \; f(x_1) = \frac{s_{\min}}{2a_1} \tag{29}$$

$$f(x_1^*) = \min_{x_1} \; f(x_1) = \frac{3}{2}s_{\min} \tag{30}$$

Note that we use the asterisk notation to denote an optimum.

Definition 1.1 (Active Constraints)**.** *When an inequality is true with equality at the optimum, then we say the inequality constraint is active. Active constraints are an important concept, and will be referenced throughout the course.*

2 Linear Programs (LP)

We are now positioned to precisely define a linear program (LP).

Definition 2.1 (Linear Program)**.** *A linear program (LP) is an optimization problem of the form (1)-(3) where the functions $f(x)$, $g(x)$, $h(x)$ are all linear in the decision variable x.*

Given the linear structure, the general form of a LP is

$$\text{Minimize:} \qquad c_1x_1 + c_2x_2 + \ldots + c_Nx_N$$

$$\begin{aligned} \text{subject to:} \quad a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,N}x_N &\leq b_1\\ a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,N}x_N &\leq b_2\\ &\vdots\\ a_{M,1}x_1 + a_{M,2}x_2 + \ldots + a_{M,N}x_N &\leq b_M. \end{aligned}$$

Alternatively, we can express this in compact form using the "Sigma" notation

$$\text{Minimize:} \qquad \sum_{k=1}^{N} c_k x_k$$

$$\begin{aligned} \text{subject to:} \quad \sum_{k=1}^{N} a_{1,k} x_k &\leq b_1 \\ \sum_{k=1}^{N} a_{2,k} x_k &\leq b_2 \\ &\vdots \\ \sum_{k=1}^{N} a_{M,k} x_k &\leq b_M \end{aligned}$$

We can further compact the notation using matrix notation

$$\text{Minimize:} \qquad c^T x$$

$$\text{subject to:} \qquad Ax \leq b$$

where

$$\begin{aligned} x &= [x_1, x_2, \ldots, x_N]^T \\ c &= [c_1, c_2, \ldots, c_N]^T \\ [A]_{i,j} &= a_{i,j}, \quad A \in \mathbb{R}^{M \times N} \\ b &= [b_1, b_2, \ldots, b_M]^T \end{aligned}$$

Exercise 2 (Building a Solar Array Farm)**.** You are tasked with designing the parameters of a new photovoltaic array installation. Namely, you must decide on the square footage of the photovoltaic arrays, and the power capacity of the power electronics which interface the generators to the grid. The goal is to minimize installation costs, subject to the following constraints:

- You cannot select negative PV array area, nor negative power electronics power capacity.
- The minimum generating capacity for the photovoltaic array is $g_{\min}$.
- The power capacity of the power electronics must be greater than or equal to the PV array power capacity.
- The available spatial area for installation is limited by $s_{\max}$.

- You have a maximum budget of $b_{\max}$.

Using the notation in Table 3, perform the following steps

1. Write down the objective function
2. Write down the constraints. Label the physical meaning of each constraint.
3. Re-write the entire mathematical optimization problem in Standard Form (see Table 2).
4. Re-write the LP in matrix-vector form, with appropriate definitions for c, A, b.

Table 3: Building a Solar Array Farm

Spatial area of photovoltaic arrays [m^2]	x_1
Power capacity of power electronics [kW]	x_2
Cost of square meter of PV array [USD/m^2]	c_1
Cost of power electronics per kW [USD/kW]	c_2
Min PV array generating capacity [kW]	$g_{\min}$
Power of PV array per area [kW/m^2]	a_1
Max spatial area [m^2]	$s_{\max}$
Maximum budget [USD]	$b_{\max}$

3 Example 1: Transportation Problem

In this section, we introduce the *transportation problem* - a typical transportation system engineering problem that can be formulated and solved as a linear program. More information about the transportation problem can be found in Chapter 6.D of [1].

Consider a supplier-buyer network consisting of two apple farms and two factories. The farm production and factory consumption rates are given by Table 4. George owns both the farms and factories. He is paying the cost of shipping all the apples from the farms to the factories. The shipping costs for George are given by Table 5.

The problem is to determine the best way to distribute apple shipments. The problem can be visualized by the network graph in Fig. 4. The left-hand nodes represent farms, the right-hand nodes represent factories, and the edges represent shipping costs. Let s_i represent the

Table 4: Supply and demand capacities for the farms and factories, respectively.

Paul's farm produces 4 tons of apples per day	$s_p = 4$
Ron's farm produces 2 tons of apples per day	$s_r = 2$
Max's factory needs 1 ton of apples per day	$d_m = 1$
Bob's factory needs 5 tons of apples per day	$d_b = 5$

Table 5: Shipping costs from farm to factory.

Paul $\rightarrow$ Max: 1000 USD per ton	$c_{pm} = 1000$	x_{pm}
Ron $\rightarrow$ Max: 1250 USD per ton	$c_{rm} = 1250$	x_{rm}
Paul $\rightarrow$ Bob: 1350 USD per ton	$c_{pb} = 1350$	x_{pb}
Ron $\rightarrow$ Bob: 1450 USD per ton	$c_{rb} = 1450$	x_{rb}

farm production supply, d_j represent the factory demand, x_{ij} represent tons of apples shipped from farm i to factory j, and c_{ij} is the corresponding cost. Indices $i \in \{p, r\}$ denote farms and $j \in \{m, b\}$ denote factories.

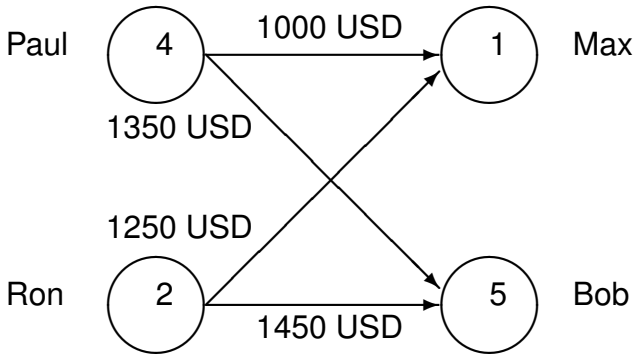

Figure 4: Apple Farm-Factory Network Graph

With this notation established, we can write the total shipping costs as

$$f = c_{pm}x_{pm} + c_{pb}x_{pb} + c_{rm}x_{rm} + c_{rb}x_{rb} \tag{31}$$

Moreover, the factory demand constraints must be satisfied

$$x_{pm} + x_{rm} = d_m, \tag{32}$$
$$x_{pb} + x_{rb} = d_b, \tag{33}$$

and the farm supply capacity must be satisfied

$$x_{pm} + x_{pb} = s_p, \tag{34}$$
$$x_{rm} + x_{rb} = s_r. \tag{35}$$

In addition, the number of shipped apples along each edge must be non-negative

$$x_{pm} \geq 0,\ x_{pb} \geq 0,\ x_{rm} \geq 0,\ x_{rb} \geq 0. \tag{36}$$

We can now assemble the cost function and constraints to produce the linear program

$$\text{min:} \quad f(x_{pm}, x_{pb}, x_{rm}, x_{rb}) = c_{pm}x_{pm} + c_{pb}x_{pb} + c_{rm}x_{rm} + c_{rb}x_{rb} \tag{37}$$

$$\begin{aligned}
\text{s. to} \quad x_{pm} + x_{rm} &= d_m && (38)\\
x_{pb} + x_{rb} &= d_b && (39)\\
x_{pm} + x_{pb} &= s_p && (40)\\
x_{rm} + x_{rb} &= s_r && (41)\\
x_{pm} \geq 0,\ x_{pb} \geq 0,\ x_{rm} \geq 0,\ x_{rb} &\geq 0 && (42)
\end{aligned}$$

3.1 General LP Formulation

Next we consider a general formulation for the transportation problem. Consider a generalized supplier-consumer network graph, consisting of N consumers and M suppliers, as shown in Fig. 5. As before, s_i represents the supply capacity of supplier i, d_j represent the demand of consumer j, x_{ij} represents the shipped goods from supplier i to consumer j, and c_{ij} is the corresponding cost.

Then we can write the transportation problem in the general LP formulation

$$\text{min:} \quad \sum_{i=1}^{M}\sum_{j=1}^{N} c_{ij}x_{ij} \tag{43}$$

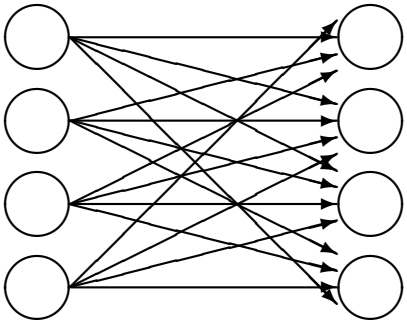

Figure 5: Supplier-Consumer network graph, consisting of N consumers and M suppliers.

$$\text{s. to} \quad \sum_{i=1}^{M} x_{ij} = d_j, \qquad j = 1, \cdots, N \tag{44}$$

$$\sum_{j=1}^{N} x_{ij} = s_i, \qquad i = 1, \cdots, M \tag{45}$$

$$x_{ij} \geq 0, \qquad \forall i, j \tag{46}$$

Exercise 3 (Transportation Problems)**.** Exercises 6.2 and 6.4 in Revelle and Whitlatch [1].

4 Example 2: Shortest Path Problem

In this section, we introduce the *shortest path problem* - a typical transportation system engineering problem that can be formulated and solved as a linear program. In fact, this exact problem is solved when searching for directions on Google Maps. More information about the shortest path problem can be found in Chapter 6.B of [1].

Consider a network of locations (nodes), connected by various routes (edges) in the network graph in Fig. 6. The problem is to find the shortest path from Alice's house (node A) to Bob's house (node B). For tutorial purposes, suppose the shortest path is the path denoted in red in Fig. 6. Let c_{ij} represent the distance from node i to node j, where $i, j \in \{A, 2, 3, \cdots, 9, 10, B\}$. Then the shortest path length is given by $c_{A3} + c_{34} + c_{45} + c_{59} + c_{9B}$.

Let us now consider how to formulate an LP that produces the shortest path solution. Let x_{ij} represent whether the edge between nodes i and j is included in the chosen path. That is, $x_{ij} = 1$ for every (i, j) on the chosen path and $x_{ij} = 0$ for every (i, j) NOT on the chosen path. For example, in the path denoted in red, $x_{A3} = x_{34} = x_{45} = x_{59} = x_{9B} = 1$ and $x_{ij} = 0$ for all other (i, j) pairs.

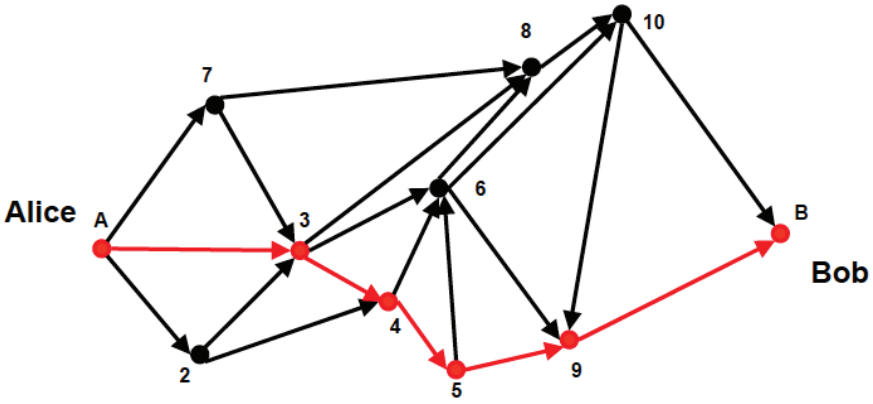

Figure 6: The shortest path problem is to determine the shortest path between Alice and Bob on the network above.

Using these notational definitions, it is straight-forward to verify

$$\sum_{(i,j) \text{ chosen on path}} c_{ij} = \sum_{(i,j) \text{ chosen on path}} c_{ij} x_{ij} = \sum_{\text{all } (i,j)} c_{ij} x_{ij}$$

As a result, we can formulate the cost function

$$J = \sum_{j \in \mathcal{D}_A} c_{Aj}\, x_{Aj} + \sum_{i=2}^{10} \sum_{j \in \mathcal{D}_i} c_{ij} x_{ij} + \sum_{i \in \mathcal{A}_B} c_{iB} x_{iB} \tag{47}$$

where sets $\mathcal{D}_A, \mathcal{D}_i$ are the subsets of nodes that descend from nodes A and i, respectively. For example, $\mathcal{D}_A = \{2, 3, 7\}$ and $\mathcal{D}_5 = \{6, 9\}$. Set $\mathcal{A}_B$ is the subset of nodes that ascend from node B, namely $\mathcal{A}_B = \{9, 10\}$. Consequently, the terms in (47) respectively denote the first leg length, intermediate leg lengths, and final leg length.

In addition to the cost function, we can formulate constraints that "stitch" the legs together and ensure they begin and end at nodes A and B, respectively. The leg stitching constraint can be interpreted as follows. If a selected path arrives to node j, then a selected path must also depart from node j. Mathematically

$$\sum_{i \in \mathcal{A}_j} x_{ij} = \sum_{k \in \mathcal{D}_j} x_{jk}, \quad j = 2, \cdots, 10 \tag{48}$$

The origin and destination constraints are given by

$$\sum_{j \in \mathcal{D}_A} x_{Aj} = 1, \tag{49}$$

$$\sum_{i \in \mathcal{A}_B} x_{iB} = 1. \tag{50}$$

Finally, we have the non-negativity constraints

$$x_{ij} \geq 0, \quad \forall\, i,j \in \{2, \cdots, 10\}, \qquad x_{Aj} \geq 0, \quad \forall\, j \in \mathcal{D}_A, \qquad x_{iB} \geq 0, \quad \forall\, i \in \mathcal{A}_B \tag{51}$$

To summarize, we can write the shortest path problem as the following LP:

Minimize: $$J = \sum_{j \in \mathcal{D}_A} c_{Aj}\, x_{Aj} + \sum_{i=2}^{10} \sum_{j \in \mathcal{D}_i} c_{ij} x_{ij} + \sum_{i \in \mathcal{A}_B} c_{iB} x_{iB} \qquad \text{[total path length]} \tag{52}$$

subject to:

$$\sum_{i \in \mathcal{A}_j} x_{ij} = \sum_{k \in \mathcal{D}_j} x_{jk}, \quad j = 2, \cdots, 10, \qquad \text{[leg stiching]} \tag{53}$$

$$\sum_{j \in \mathcal{D}_A} x_{Aj} = 1, \qquad \text{[origin]} \tag{54}$$

$$\sum_{i \in \mathcal{A}_B} x_{iB} = 1, \qquad \text{[destination]} \tag{55}$$

$$x_{ij} \geq 0, \quad \forall\, i,j \in \{2, \cdots, 10\}, \qquad \text{[middle node non-negativity]} \tag{56}$$

$$x_{Aj} \geq 0, \quad \forall\, j \in \mathcal{D}_A, \qquad \text{[origin node non-negativity]} \tag{57}$$

$$x_{iB} \geq 0, \quad \forall\, i \in \mathcal{A}_B. \qquad \text{[destination node non-negativity]} \tag{58}$$

5 Graphical Solutions to LPs

For problems of one, two, or sometimes three dimensions, we can use graphical methods to visualize the feasible sets and solutions. This visualization provides excellent intuition for the nature of LP solutions. Moreover, the graphical solution process motivates the concepts of **feasible set**, **boundedness**, **uniqueness**, and **constraint domination**.

5.1 Feasible Set

Consider the following example.

$$\max \qquad Z = 140x_1 + 160x_2 \tag{59}$$

$$\text{s. to} \quad 2x_1 + 4x_2 \leq 28 \tag{60}$$
$$5x_1 + 5x_2 \leq 50 \tag{61}$$
$$x_1 \leq 8 \tag{62}$$
$$x_2 \leq 6 \tag{63}$$
$$x_1 \geq 0 \tag{64}$$
$$x_2 \geq 0 \tag{65}$$

On a graph, one may successively plot each of the inequality constraints (60)-(65) and divide the Cartesian space into feasible "half-spaces." This procedure is demonstrated in the left side of Fig. 7. As each half-space is identified, we retain the intersection of the remaining feasible set. This successive construction provides the **feasible set**, as shown in Fig. 7. This intuition motivates the following formal definition of **feasible set**.

Definition 5.1 (Feasible Set)**.** *Consider the canonical optimization problem (1)-(3). The feasible set $\mathcal{D}$ is the set of all possible designs x that satisfy the constraints, i.e. $\mathcal{D} = \{x \in \mathbb{R}^n \mid g(x) \leq 0,\ h(x) = 0\}$.*

After constructing the feasible set, we plot the iso-contours of the objective function. For example, the lower-left-most iso-contour in the right side of Fig. 7 corresponds to $Z = 0$. Continuing towards the upper-right, the value of the objective function increases. The intersection of the largest-valued iso-contour and the feasible set occurs when $Z^* = 1480$, at $x_1^* = 6, x_2^* = 4$. Consequently, we have graphically solved the LP.

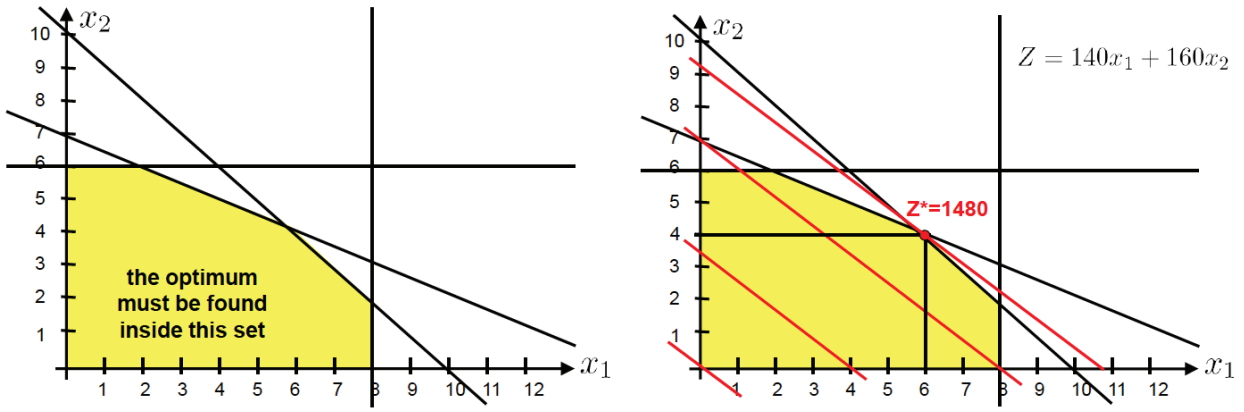

Figure 7: Construction of the feasible set in a linear program [LEFT], and the objective function isolines [RIGHT].

5.2 Boundedness

A feasible set can fall within one of the following three categories:

- **[Bounded]** The feasible set is bounded if it forms a closed subset of the Cartesian plane that does not include infinity.
- **[Unbounded]** The feasible set is unbounded if it forms a subset of the Cartesian plane that includes infinity.
- **[Empty]** The feasible set is empty if the intersection of all the constraints forms the empty set. In this case the problem is infeasible. That is, no solution exists.

Exercise 4. *Draw examples of each of the three categories given above.*

Also note that an objective function may be bounded or unbounded. We make these concepts concrete with the following examples. Consider the feasible set defined by inequalities $x_2 \geq 3$, $x_1 + x_2 \geq 6$, and $x_2 \leq 0.5x_1 + 7$, as shown in Fig. 8. On the left, consider the objective $\max Z = 3.24x_1$. The iso-contours continue towards $x_1 = \infty$, without being bounded by the feasible set. Hence, objective function Z is unbounded. In contrast, consider the objective $\min Z = x_1 + 3x_2$. Although the feasible set is unbounded, the iso-contours are bounded as they decrease in value. In this case, objective function Z is bounded.

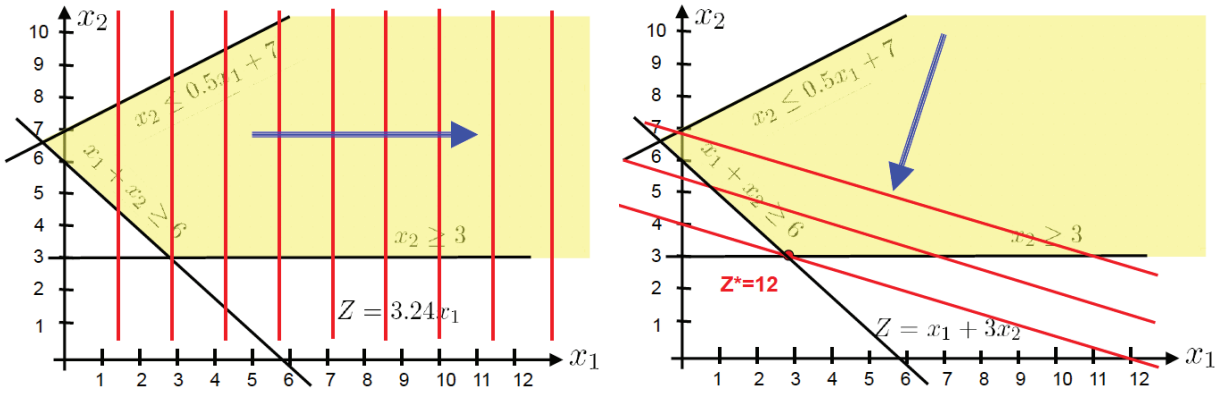

Figure 8: An unbounded [LEFT] and bounded [RIGHT] objective function.

5.3 Solution Uniqueness

This graphical analysis motivates the following proposition about LP solutions.

Proposition 2 (LP Solutions)**.** The solution to any linear program is characterized by one of the following three categories:

- **[No Solution]** This occurs when the feasible set is empty, or the objective function is unbounded.
- **[One Unique Solution]** There exists a single unique solution at the vertex of the feasible set. That is, two constraints are active and their intersection gives the optimal solution.
- **[A Non-Unique Solution]** There exists an infinite number of solutions, given by one edge of the feasible set. That is, one constraint is active and all solutions along this edge are equally optimal. This can only occur when the objective function gradient is orthogonal to a constraint.

Exercise 5. *Construct graphical examples of each of the three possible LP solutions given above.*

5.4 Constraint Domination

Consider a LP with the inequality constraints

$$x_1 \geq 2, \tag{66}$$

$$x_1 \geq 4. \tag{67}$$

Figure 9 draws these constraints on the $x_1 - x_2$ plane and divides the Cartesian space into half-spaces. In this case, we note that constraint $x_1 \geq 4$ **dominates** $x_1 \geq 2$. That is, $x_1 \geq 2$ is automatically satisfied when $x_1 \geq 4$ is satisfied. As a result, we can disregard the dominated constraint $x_1 \geq 2$. This concept of **constraint domination** enables the optimization engineer to reduce the number of constraints.

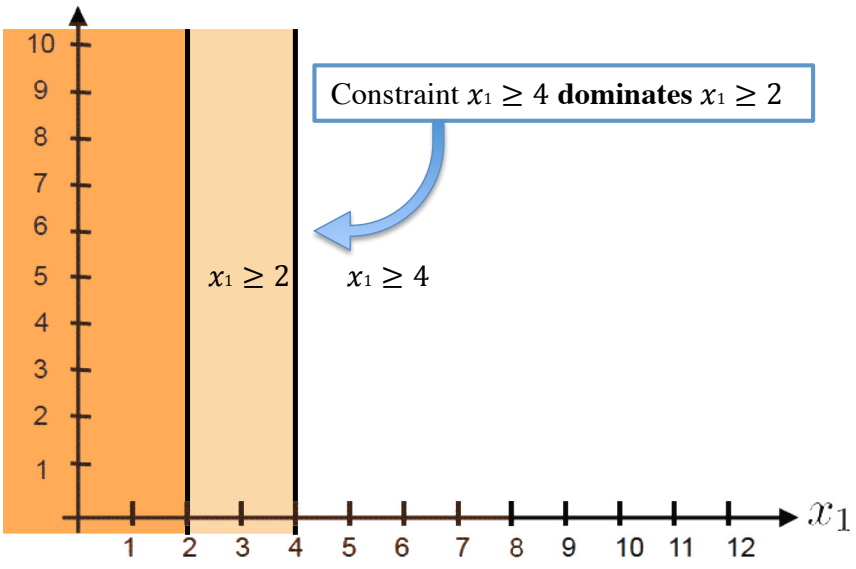

Figure 9: The constraint $x_1 \geq 2$ is dominated by $x_1 \geq 4$.

5.5 A General Method for Graphical LP Solutions

Armed with the concepts of feasible sets, boundedness, and uniqueness, we now present a flow chart of graphical LP solutions in Fig. 10. Note this procedure is useful for problems of one, two, or sometimes three dimensions. However, real-world civil engineering problems are typically have hundreds, thousands, or even millions of dimensions. As a result, an automated procedure is necessary.

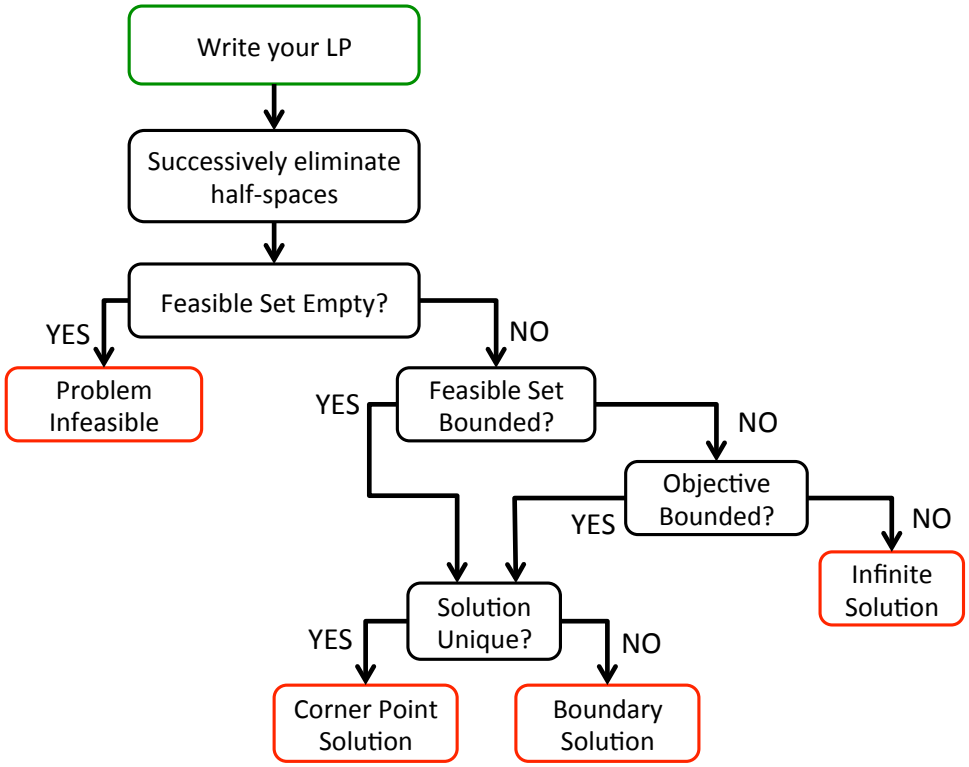

Figure 10: Flowchart for graphically solving LPs.

Exercise 6 (Graphical LP)**.** Exercises 3.1 to 3.12 in Revelle and Whitlatch [1].

6 Simplex Algorithm

George B. Dantzig's Simplex Algorithm is an automated procedure for solving LPs [2]. It stands as one of the most significant algorithmic achievements of the 20th century. Danzig (1914 - 2005) was a Professor Emeritus of Transportation Sciences and Professor of Operations Research and of Computer Science at Stanford.

In this section we provide an overview of the simplex algorithm concepts without a deep exposition on the mathematical implementation. Today, the simplex algorithm is a mature off-

the-shelf technology. As such, the modern engineer will rarely implement the algorithm themselves. Nonetheless, the intuition behind the algorithm is instructive, as it builds upon the concepts throughout the chapter.

As shown in Proposition 2, LPs can have one of three types of solutions: no solution (in feasible or infinite), a unique solution (a corner point), or a non-unique solution (a boundary). The basic idea of the simplex method is to confine the search to corner points of the feasible region (of which there are only finitely many) in a most intelligent way. Consider a LP in standard form

$$\text{Minimize:} \qquad c^T x \tag{68}$$

$$\text{subject to:} \qquad Ax \leq b, \tag{69}$$

$$A_{eq}x = b_{eq} \tag{70}$$

In geometric terms, the feasible set $\mathcal{D} = \{x \mid Ax \leq b,\ A_{eq}x = b_{eq}\}$ is a (possibly unbounded) convex polytope, which can be characterized by its corner points. Proposition 2 says if the objective function has a minimum value on the feasible set, then the minimum occurs on (at least) one of the corner points. This reduces the problem to a finite computation since there is a finite number of corner points, but the number of corner points is unmanageably large for all but the smallest linear programs.

It can also be shown that if a corner point is not a minimum point of the objective function then there is an edge containing the point so that the objective function is strictly decreasing on the edge moving away from the point. If the edge is finite then the edge connects to another corner point where the objective function has a smaller value, otherwise the objective function is unbounded below on the edge and the linear program has no solution. The simplex algorithm applies this insight by walking along edges of the polytope to corner points with lower and lower objective values. This continues until the minimum value is reached or an unbounded edge is visited, concluding that the problem has no solution. The algorithm always terminates because the number of vertices in the polytope is finite; moreover since we jump between vertices always in the same direction (that of the objective function), we hope that the number of vertices visited will be small.

7 Notes

You can learn more about the essential elements of optimization models in Ch. 1 of Papalombros and Wilde [3]. An excellent introduction of mathematical optimization formulations is provided in Ch. 1 of Boyd and Vandenverghe [4]. An overview of CEE system optimization applications are given in Ch. 2 of Revelle and Whitlatch [1].

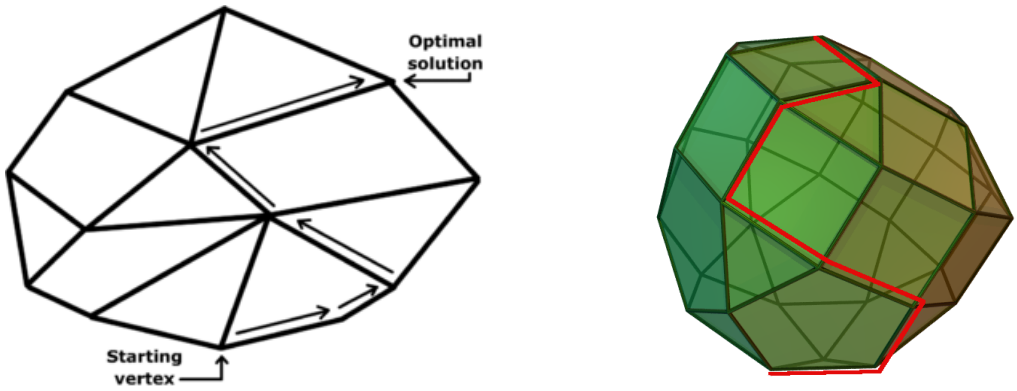

Figure 11: A system of linear inequalities defines a polytope as a feasible region. The simplex algorithm begins at a starting vertex and moves along the edges of the polytope until it reaches the vertex of the optimum solution.

The textbook by Dantzig [2] is a classic reference for linear programming. Chapter 6 of Revelle and Whitlatch [1] contains many LP models of network flow, including the shortest path problem, the transportation problem, the transshipment problems, the maximum flow problem, and the traveling salesmen problem. Graphical solution methods for LPs are discussed in Ch. 3 of Revelle and Whitlatch [1]. Finally, Ch. 4 of Revelle and Witlach [1] and Dantzig [2] provide excellent expositions of the Simplex Algorithm.

References

[1] C. A. Revelle and E. E. Whitlach, Civil and environmental systems engineering. Prentice Hall PTR, 1996.

[2] G. B. Dantzig, Linear programming and extensions. Princeton university press, 1998.

[3] P. Y. Papalambros and D. J. Wilde, Principles of Optimal Design: Modeling and Computation. Cambridge University Press, 2000.

[4] S. Boyd and L. Vandenberghe, Convex optimization. Cambridge university press, 2009.