Mathematical Optimization

Zac Zerafa

November 12, 2025

Contents

Part I

Fundamentals

Chapter 1

Optimization problems

Mathematical optimization seeks to apply mathematics to real life problems to determine the most efficient allocation of 'resources' (whether that be physical resources, time, or any other abstraction of the term). The nature of mathematical optimization may differ depending on the type of problem; the subfield of linear programming (LP) draws extensively from convex analysis and linear algebra, however nonlinear integer programming would require a deeper understanding of combinatorics and number theory.

Nevertheless, the fundamentals of mathematical optimization pays some homage to a range of definitions and concepts, most of which lie at the interplay of elementary set theory, order theory, and algebra.

Before one even considers the use of mathematics to solve optimization problems, they would have to resort to a framework to define their problem in a mathematical sense. We start by developing the tools that can describe various rules in which a resource may legally be allocated.

Definition 1.1. A *dependent variable (DV)* x is a variable representing some quantity that can be changed. It often represents an amount of some 'resource' available to us.

We will often use a vector $\mathbf{x}$ to represent the vector with the dependent variables as entries.

Often we want to impose rules on which values dependent variables can have. For instance, perhaps x_1 should never be less than x_2.

Definition 1.2. The *feasible region* X is the set of all permissible combinations of values that the DVs may have. Elements of the feasible region are called *feasible solutions*.

The DVs essentially provide a way of quantifying the thing we seek to optimally distribute, and feasible regions

It's crucial to have some method to determine how optimal elements of the feasible region really are. This method is represented as a function, and its its minimum or maximum.

Definition 1.3. An *objective function (OF)* $f : X \to Y$ is a function whose domain is restricted to the feasible region. This function determines a way to compare how 'desirable' different feasible solutions are.

Definition 1.4. A *goal function (GF)* $g : F \to \mathbb{R}$ is a function that maps OFs to the 'most desirable' element in its image. Commonly, these are either the min or max functions.

Definition 1.5. An *optimization problem* is a 3-tuple (X, f, g) of a feasible region$X$, an objective function f with X as its domain, and a 'goal' set function g that states how to determine the optimal solution from the objective function (min or max). Problems with the min goal function are minimization optimization problems, while those endowned with the max goal function are maximization optimization functions.

We now define what constitutes a solution to an optimization problem.

Definition 1.6. The *optimal solution* of an optimization problem (X, f, g) is the following represented by z.

$$z = g\{f(x) : x \in X\}$$

Definition 1.7. An *optimal parameter* of an optimization problem (X, f, g) is an element x_0 in the feasible region that optimizes the OF.

$$\mathbf{x}_0 \in X \text{ is an optimal parameter of } (X, f, g) \iff f(\mathbf{x}_0) = g_{\mathbf{x}\in X}(f)\}$$

Note that the optimal parameter and solution may not exist (i.e be undefined or unbounded; we will get to these soon), making someoptimization problems degenerate.

One can show a special relationship between the min and max goal functions that essentialy implies that the two types of optimization problems can trivially be converted from one to another.

Theorem 1.1 (Min-Max equivalence)**.** Given a minimization optimization problem $(X, f, \min)$,the maximization problem $(X, -f, -\max)$ has the same optimal solution and optimal parameters.

This claims that minimization and maximization differs only by a sign. Because of this result, the majority of this book will discuss and present theory based on minimization optimization functions.

Now let's discuss the rather 'dissapointing' types of optimization problems.

Definition 1.8. A minimization optimization problem is *infeasible* iff its feasibility region is the null set.

$$(X, f, g) \text{ is infeasible } \iff X = \emptyset$$

It makes no sense to discuss how to execute an impossible task 'optimally'; the notion of infeasibility captures this.

Definition 1.9. An optimization problem is *unbounded* iff for any element of the feasible region, one can always find a more optimal element in the feasible region.

$$(X, f, \min) \text{ is unbounded } \iff \forall x \in X(\exists y \in X[f(y) < f(x)])$$

$$(X, f, \max) \text{ is unbounded } \iff \forall x \in X(\exists y \in X[f(y) > f(x)])$$

It makes no sense to discuss how to execute an impossible task 'optimally'; the notion of infeasibility captures this.

The notion of an optimization problem pretty much defines the scope of this branch of mathematics; if some dilemma is reducible to an optimization problem, the theory of this field may prove useful.

Granted, this definition of an optimization problem is quite general and indeed encompasses many trivial problems, impossible problems, and even problems defined in such a way thatone can do no better than a brute force algorithm on its feasible region.

Here is a basic optimization problem to build some inuition with the mathematical notation.

Example 1.1. John is 185cm tall and Jane is 167cm tall. A pair of Yeezys add 3cm to one's height and a pair or Air Forces add 5cm. Which combination of person and pair of shoes is closest to 176cm, given that shoes must be worn?

$$f(\mathbf{x}) = |176 - x_1 + x_2|$$

$$X = \{\begin{bmatrix}185\\3\end{bmatrix}, \begin{bmatrix}185\\5\end{bmatrix}, \begin{bmatrix}167\\3\end{bmatrix}, \begin{bmatrix}167\\5\end{bmatrix}\}$$

$$z = \min_{\mathbf{x}\in X}(f) = 4$$

$$\mathbf{x}_0 = \text{argmin}_{\mathbf{x}\in X}(f) = \begin{bmatrix}167\\5\end{bmatrix}$$

This problem in itself is kind of dull (there could be worse problems, imagine a problem with only one element in its feasible region!), however this notation can also express optimization problems in much more complicated contexts.

Chapter 2

Constraints

Let's consider the following example.

Example 2.1. Enter a God awful example here with 3 inequalities

This example does something pretty interesting; it defines its feasible region with respect to 3 different inequalities which all have to be satisfied, or in the fancy language of set theory and mathematical logic, it defines the feasible region by using a conjunction of predicates in the setbuilder logic.

Definition 2.1. A *constraint* is some condition (technically called a 'predicate') on the dependent variables. These can be used to define types of feasible regions so that all feasible solutions must satisfy each constraint.

One can take the notion of a constraint to create extremely bizzare (and even ill-posed) optimization problems, however inequalities and equalities are by far the most common type of constraint; the use of minimum and maximum functions as well as the fact that inequalities are widely studied constraints is precisely why order theory has so much to bestow upon mathematical optimization.

Part II

Linear Programming

Chapter 3

Linear programs

A class of optimization problems that tackle a decent range of real life applications are linear programs. Indeed, linear programs form the basic theory for more complex optimation problems.

Definition 3.1. A *linear program* (LP) is an optimization problem 'equivalent' to some $(X, f, \min)$, with matrix $\mathbf{A}$ and vectors $\mathbf{b}, \mathbf{c}$ such that:

$$X = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} \geq \mathbf{b} \wedge \mathbf{x} \geq \mathbf{0}\}$$

Similarly, for $(X, f, \max)$ we have the following.

$$X = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} \leq \mathbf{b} \wedge \mathbf{x} \geq \mathbf{0}\}$$

$$f(\mathbf{x}) = \mathbf{c}^\intercal \mathbf{x}$$

Since this matrix and pair of vectors can define a linear program, the 4-tuple $(\mathbf{A}, \mathbf{b}, \mathbf{c}, g)$ may be used to refer to LPs.

- $\mathbf{A} : n \times m$ is the constraint matrix
- $\mathbf{b} : m$ is the constraint vector
- $\mathbf{c} : n$ is the cost vector

The use of the word 'equivalent' roughly means that the feasible region, optimal solution and optimal parameters are the same as some optimization problem constructed by the definition of an LP. We will delve into this idea of LP equivalence soon.

We are already familiar with some basic LPs. Take the example from the last chapter; we can formally represent it as an LP in the following way. - represent the optimization prob as an LP here

Often it is the case where a real-life problem has dependent variables that are not restricted in sign (i.e they violate $\mathbf{x} \geq 0$). Fortunately, thefollowing fact can be used to transform it into the standard form for an LP.

Proposition 3.1. Any real number is the difference of two nonnegative real numbers.

$$x \in \mathbb{R} \iff \exists x_1, x_2 \in [0, \infty)[x = x_1 - x_2]$$

3.1 Graphical method

- Change all inequality constraints to

3.2 Canonical form

In our definition, the condition we imposed on our matrix and vectors to form the feasible region is called the *normal form of an LP*. Indeed, one can define LPs with respect to different 'forms' which are 'equivalent' to the normal form; every LP in normal form has an 'equivalent' LP with respect to another form (note that this LP may use a different matrix than $\mathbf{A}$ and vector than $\mathbf{b}$); we define equivalence inthe following manner.

Definition 3.2. Let (X, f, g) and (Y, f, h) be two LPs with the same OF, then the pair of LPs is an *pair of equivalent LPs* iff the feasible regions X, Y form the same set and the LPs have the same optimal solution and optimal parameters.

An extremely useful form to deal with LPs is the *canonical form*; it will be required for an algorithm that we will investigate later.

Definition 3.3. An LP (X, f, g) is in *canonical form* iff

$$X = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} = \mathbf{b} \wedge \mathbf{x} \geq \mathbf{0} \wedge \mathbf{b} \geq \mathbf{0}\}$$

The proof equating canonical and normal forms is omitted, however it will be a simple consequence of the techniques listed in this chapter. Note that

its use of equalities allows for a much easier analysis since direct methods from linear algebra can be applied!

However how does one convert inequalities of the normal form to equalities in the canonical form? Any inequality can be made into an equality, albeit at the cost of introducing new nonnegative variables. These variables introduced are called *slack and excess variables*.

Proposition 3.2. If for two nonnegative variables we have $x \leq y$, a nonnegative *slack variable* may be added to x to acheive equality.

$$x, y \in [0, \infty) \implies [x \leq y \iff \exists s \in [0, \infty)(x + s = y)]$$

If for two nonnegative variables we have $x \geq y$, a nonnegative *excessvariable* may be subtracted to x to acheive equality.

$$x, y \in [0, \infty) \implies [x \geq y \iff \exists e \in [0, \infty)(x - e = y)]$$

Slack variables are dummy variables used to represent remaining addable quantity to reach equality. Surplus variables are dummy variables that represent excess quantity from equality.

3.3 Application of convex analysis

Now that we are able to convert any problem to canonical form, we turn to some results from convex analysis that affect the field of linear programming.

Proposition 3.3. For any LP (X, f, g), X is convex,

$$(X, f, g) \text{ is an LP} \implies X \text{ is convex}$$

Proposition 3.4. feasible sets have optimal solution at an extreme point feasible sets have finite extreme points

These results mean that if any extreme point can be calculated, any LP can theoretically be solved by calculating the finite amount of extreme points, passing them through the OF, and taking the most optimal value of this set. Hence extreme points in an LP are called *basic feasilbe solutions*, due to their potential of being a solution by the proposition.

Definition 3.4. *basic feasible solution* (BFS) of an LP is an extreme point of its feasible region.

Unfortunately, larger problems have astronomical amounts of BFSs; for n variables and m constraints, one has $\binom{n}{m}$ BFSs in the worst case(depending on factors such as linear dependence etc.). For a practical algorithm, one must reduce the amount of BFSs involved.

Chapter 4

Simplex method

We shall develop a greedy algorithm that starts with some initial BFS and seeks an 'adjacent' BFS that is 'more optimal'. Consider m constraints and n variables, where $n \geq m$. Then we have $n - m$ redundant variables that may be set to zero. The remaining m variables are called the *basis elements*, let them form the set B. It is most conventient to make the initial BFS that where the slack variables are the basis elements. For a given basis B, let us define the following notations:

- $\mathbf{B}$ is the matrix of columns of $\mathbf{A}$ corresponding to basis elements
- $\mathbf{N}$ is the matrix of columns of $\mathbf{A}$ corresponding to non-basis elements
- $\mathbf{x}_B$ is the vector of variables corresponding to basis elements (which we set as zeros)
- $\mathbf{x}_N = \mathbf{0}$ is the vector of variables corresponding to non-basis elements
- $\mathbf{c}_B$ is the vector of cost coefficients corresponding to basis elements
- $\mathbf{c}_N = \mathbf{0}$ is the vector of cost coefficients corresponding to non-basis elements

Adjacent BFSs are those that differ by a single variable; our algorithm should jump to an adjacent BFS by determining the most optimal nonbasis variable to enter the BFS and utilize this variable until it renders another basis element equal to 0, essentially booting it from our BFS.

4.1 Calculating arbitrary BFS

Assume that our LP is in canonical form, feasible solutions are solutions to this equality.

$$\mathbf{Ax} = \mathbf{b}$$

Since A is not necessarily square (so there may not exist an $\mathbf{A}^{-1}$), we consider breaking it up in the following manner.

$$\mathbf{B}\mathbf{x}_B + \mathbf{N}\mathbf{x}_N = \mathbf{b}$$

Recall that we set $\mathbf{x}_N = \mathbf{0}$

$$\mathbf{B}\mathbf{x}_B = \mathbf{b}$$

$$\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b}$$

Hence the solution of the current BFS:

$$\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b}$$

4.2 Determining 'optimal' variable

In order to consider which nonbasis element is ideal to add to the BFS, we consider rewriting the OF to include the nonbasis elements. Although we have officially set all these nonbasis elements to 0, it's inclusion in the OF gives algebraic hints as to which variable would improve our solution if we were to include it in the BFS. To begin along this train of though, we start by rewriting basis elements with respect to non-basis elements.

$$\mathbf{B}\mathbf{x}_B + \mathbf{N}\mathbf{x}_N = \mathbf{b}$$

$$\mathbf{x}_B = \mathbf{B}^{-1}(\mathbf{b} - \mathbf{N}\mathbf{x}_N)$$

$$\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N$$

Now to consider what the OF results to

$$z = \mathbf{c}^\intercal\mathbf{x}$$

$$z = \mathbf{c}_B^\intercal\mathbf{x}_B + \mathbf{c}_N^\intercal\mathbf{x}_N$$

$$z = \mathbf{c}_B^\intercal(\mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N) + \mathbf{c}_N^\intercal\mathbf{x}_N$$

$$z = \mathbf{c}_B^\intercal \mathbf{B}^{-1}\mathbf{b} - \mathbf{c}_B^\intercal \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N + \mathbf{c}_N^\intercal \mathbf{x}_N$$
$$z = \mathbf{c}_B^\intercal \mathbf{B}^{-1}\mathbf{b} - (\mathbf{c}_B^\intercal \mathbf{B}^{-1}\mathbf{N} - \mathbf{c}_N^\intercal)\mathbf{x}_N$$
$$z + (\mathbf{c}_B^\intercal \mathbf{B}^{-1}\mathbf{N} - \mathbf{c}_N^\intercal)\mathbf{x}_N = \mathbf{c}_B^\intercal \mathbf{B}^{-1}\mathbf{b}$$

We define the addend to z as the *reduced cost vector*. Notice that although $\mathbf{x}_N = \mathbf{0}$, we could hypothetically 'switch on' a nonbasis element to the basis to potentially minimize futher. How do we choose which element will be beneficial to add to the basis? The one associated with the largest coefficient in the reduced cost vector is the ideal candidate; this is because the larger the coefficient, the larger a value will be subtracted from our current 'solution'. This gives us a method to determine which nonbasis element it would be most desirable to 'switch on'.

$$\hat{\mathbf{c}^\intercal} = \mathbf{c}_B^\intercal \mathbf{B}^{-1}\mathbf{N} - \mathbf{c}_N^\intercal$$

4.3 Ratio test

Let x_j be the entering variable. Now that we know that the larger we make x_j, the better our 'solution' will become; this is essentially sending us greedily to the next BFS. We want to make x_j large enough until the constraints force one of the basis elements to become 0; this is the sign that tells us that we have reached the next extreme point (remember; optimal solutions are at extreme points). To do this, we analyze this previously derived formula linking the basis and nonbasis variables.

$$\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N$$

We are trying to find out which basis element is the first to hit 0 when increasing x_j, so we can set $\mathbf{x}_B = \mathbf{0}$ and look for the smallest increase in x_j until one basis element concedes to zero; this is the basis element leaving the basis.

$$\mathbf{0} = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N$$
$$\mathbf{0} = \hat{\mathbf{b}} - \hat{\mathbf{N}}\mathbf{x}_N$$
$$\hat{\mathbf{b}} = \hat{\mathbf{N}}\mathbf{x}_N$$

We then analyze each basis element individually to see which one will be the first to buckle down to 0.

$$\hat{\mathbf{b}}_i = \hat{\mathbf{A}_{ij}} x_j$$

$$\frac{\hat{\mathbf{b}}_i}{\hat{\mathbf{A}}_{ij}} = x_j$$

This formula states how large we can make x_j until basis element x_i becomes 0. Therefore the first x_i to become 0 is the one associated with the smallest x_j.

This is known as the *ratio test*; it is used to determine which variable leaves. The polished result is as follows.

$$i = \text{argmin}_i\{\frac{\hat{\mathbf{b}}_i}{\hat{\mathbf{A}}_{ij}} : \hat{\mathbf{A}}_{ij} > 0\}$$

Now that we know how to start an LP at an initial BFS, and jump between BFS' in an efficient way, we can synthesize this information into a concrete algorithm.

4.4 Simplex method

We shall now formally define the process of the simplex method. Here is the sketch of what the algorithm should do.

- Set initial BFS where slack variables are the only nonzero variables
- Add most helpful variable to BFS (allow it to be nonzero in new BFS)
- Drop least helpful variable from BFS (force it to be zero in new BFS)
- Repeat until optimal solution acquired

The simplex method below updates the basis set B and the basis matrix $\mathbf{B}$ according to the Simplex algorithm for a minimization LP in standard form.

This is relatively straightforward to program on a modern computer, however the linear algebra required can be quite demanding if all components are calculated directly.

A *simplex method tableau* is often used for human computers so that instead of directly doing operations such as inverting $\mathbf{B}$ and matrix multiplication, one finds only the necessary row reduction steps required to preserve the following.

---

while $\max_i\{(\mathbf{c}_\mathbf{B}^\intercal\mathbf{B}^{-1}\mathbf{N} - \mathbf{c}_\mathbf{N}^\intercal)_i\} > 0$ **do**
  $e \leftarrow \text{argmax}_i\{(\mathbf{c}_\mathbf{B}^\intercal\mathbf{B}^{-1}\mathbf{N} - \mathbf{c}_\mathbf{N}^\intercal)_i\}$
  $l \leftarrow \text{argmin}_i\{\frac{(\mathbf{B}^{-1}\mathbf{b})_i}{(\mathbf{B}^{-1}\mathbf{A})_{ie}} : (\mathbf{B}^{-1}\mathbf{A})_{ie} > 0\}$
  $B \leftarrow B \cup \{x_e\}$
  $B \leftarrow B \setminus \{x_l\}$
  Update $\mathbf{B}$ as columns of $\mathbf{A}$ corresponding to elements of B
end while
$\mathbf{x_B} \leftarrow \mathbf{B}^{-1}\mathbf{b}$
$z \leftarrow \mathbf{c}_\mathbf{B}^\intercal\mathbf{x_B}$

- Ensure columns of $\hat{\mathbf{A}}$ relating to basis elements are colums of identity matrix
- Ensure indexes of $\hat{\mathbf{c}^\intercal}$ relating to basis elements are equal to 0
- Ensure simplex method tableau is in *cannonical form*

- simplex method tableau - canonical form simplex method tableau

Chapter 5

Simplex method revisions

5.1 Artificial variables

5.2 Big M simplex method

- Arrange LP into canonical form
- Add an artificial variable a_i to each constraint without a slack variable
- Update OF to $z = \mathbf{c}^\intercal \mathbf{x} + \sum_i Ma_i$ (where M is treated as a 'large' constant)
- Solve the new LP with the simplex method

5.3 Two phase simplex method

- Arrange LP into canonical form
- Add an artificial variable a_i to each constraint without a slack variable
- Solve the LP using the same constraints but an OF of $w = \sum_i a_i$
- If $w = 0$, drop the nonbasic a_i, reintroduce the OF as $z = \mathbf{c}^\intercal \mathbf{x}$ and continue solve using the simplex method
- If $w \neq 0$, the LP is infeasible

5.4 Revised simplex method

Manipulating the simplex method tableau is actually implicitly calculating $\mathbf{B}^{-1}$ for the new BFS, and the tabelau updates how $\hat{\mathbf{A}}, \hat{\mathbf{b}}, \hat{\mathbf{c}^\intercal}$ based on this new $\mathbf{B}$.

The columns of $\hat{\mathbf{A}}$ corresponding to the initial BFS actually equals the latest $\mathbf{B}^{-1}$. This is because the basis elements have columns of the identity matrix, and the initial BFS columns were originally the columns of the identity matrix, therefore the must have been multiplied by $\mathbf{B}^{-1}$.

Chapter 6

Duality theory

Every LP is mathematically associated with some other 'similar' LP called is 'dual problem'.

Definition 6.1. Given an LP $(\mathbf{A}, \mathbf{b}, \mathbf{c}, \min)$, $\mathbf{A}, \mathbf{c}, \mathbf{b}, \max)$ is called its *dual LP* and the original LP is called the *primal LP*.

Dual LPs might arise naturally in application due to having two related LPs that work with the same values, however there is much theory that can be stated about how the two LPs relate to one another in a mathematical sense.

- optimality gap -The *duality gap* of an LP is the difference between the optimal solutions of a prime LP and its dual LP. We will later see that this gap iz

Lemma 6.1. The dual of the primal's dual is the primal.

This is basically because swapping $\mathbf{b}, \mathbf{c}$ in the 4-tuple an even amount of times leaves the LP unchanged from the primal, swapping $\min, \max$ an even amount of times does the same thing.

6.1 Asymmetric dual problems

When $\mathbf{A}$ is not square, the dual will have more (or less) constraints and less (or more) variables. This may cause issues with the LP being able to be stated in canonical form; however there is a following trick to relieve the burden of cleaning up a dual problem into canonical form.

- nonnegative primal variables means dual constraint in normal form
- nonpositive primal variables means dual inequality in constraint is opposite that of the normal form
- Unrestricted primal variables mean equality in dual constraint.

After applying this proposition, one can use the familiar tricks of transforming problems into canonical form to finish the job!

6.2 Theorems of duality

Theorem 6.1 (Weak Duality Theorem; WDT)**.** Let $P = (\mathbf{b}, \mathbf{A}, \mathbf{c}, g)$ be a primal problem and D the dual of P. If $\mathbf{x}$ and $\mathbf{y}$ are feasible for P and D respectively, then the following inequality holds.

$$\mathbf{c}^\intercal \mathbf{x} \geq \mathbf{b}^\intercal \mathbf{y}$$

Corollary 6.1. Let $P = (\mathbf{b}, \mathbf{A}, \mathbf{c}, g)$ be a primal problem and D the dual of P. If P is unbounded, then D is infeasible.

Theorem 6.2 (Strong Duality Theorem for Linear Programs; SDT)**.** Let $P = (\mathbf{b}, \mathbf{A}, \mathbf{c}, \min)$ be a primal problem and D the dual of P. $\mathbf{x}$ and $\mathbf{y}$ are optimal for P and D respectively iff the following equality holds.

$$\mathbf{c}^\intercal \mathbf{x} = \mathbf{b}^\intercal \mathbf{y}$$

This result is quite powerful; it essentially says the dual LP has the same optimal solution as its primal! When constraints are mixed and the LP seems generally difficult, one technique is to *use the simplex method on the dual instead of the primal*. The SDT ensures us that we will get the same optimal solution as the primal, however the only thing that remains is the optimal parameters; how can we revover them?

Assume that using SDT we have just recovered the optimal solution z^*, we're reduced only to the constraints $z^* = \mathbf{c}^\intercal \mathbf{x}$ and the primal's constraints regarding $\mathbf{A}$ and $\mathbf{b}$. From here, we could use slack and surplus variables to make all constraints equalities and solve using Gaussian-Jordan elimination!

But we're mathematicians; the chances are that we're too lazy to deal with matrixes after all that simplex method (that's assuming you were bothered

enough to do it by hand and not delegate the task to a computer). There is a way to reduce the final set of constraints significantly, and it is by the grace of the *complementary slackness theorem* (which really should be thought of as a corollary of SDT rather than a theorem).

Theorem 6.3 (Complementary Slackness Theorem; CST)**.** Let $P = (\mathbf{b}, \mathbf{A}, \mathbf{c}, \min)$ be a primal problem and D the dual of P. If $\mathbf{x}$ and $\mathbf{y}$ are optimal for P and D respectively, then the following equality holds.

$$(\mathbf{c}^\intercal - \mathbf{y}^\intercal \mathbf{A})\mathbf{x} = 0 \wedge \mathbf{y}^\intercal(\mathbf{b} - \mathbf{A}\mathbf{x}) = 0$$

$$\mathbf{x} \text{ optimizes } P \wedge \mathbf{y} \text{ optimizes } D \iff (\mathbf{c}^\intercal - \mathbf{y}^\intercal \mathbf{A})\mathbf{x} = 0 \wedge \mathbf{y}^\intercal(\mathbf{b} - \mathbf{A}\mathbf{x}) = 0$$

Perhaps this 'theorem' initially seems like a jungle of algebra that feels intuitive, but useless. However, this theorem gives the insight to the following corollary.

Corollary 6.2. Let $\mathbf{y}$ be the optimal parameter for the dual and $\mathbf{A} : n \times m$ be the primal constraint matrix for the primal. $\mathbf{y}_i \neq 0$ iff $\sum_{k=1}^{n} \mathbf{A}_{ik}\mathbf{x}_i = \mathbf{b}_i$, where $\mathbf{x}$ is the primal optimal parameter. Essentially the ith constraint becomes an equality; this is knwon as a constraint becoming 'active'.

Using this fact, we only need to solve the system of equations including these 'active' equality constraints and $z^* = \mathbf{c}^\intercal\mathbf{x}$!

6.3 Dual simplex method

As previously mentined, we can leverage our duality theorems to solve a primal by solving the dual with the simplex method, and applying SDT and CST to reduce the problem to a (relatively) small system of linear equations. However there is yet another way that duality theory can be used to solve LPs.

The simplex method tries to iteratively optimize a BFS while keeping in the feasible region, however since feasibility of a primal problem is equivalent to feasibility of a dual problem, could one iterate a simplex method that ensures dual feasibility?

This is known as the *dual simplex method*; it works almost identically to the simplex method, except that one finds the leaving variable first using $\hat{\mathbf{b}}$ and the entering variable second using the ratio test with $\hat{\mathbf{c}^\intercal}$.

$$i = \text{argmin}\{|\frac{\mathbf{c}_j}{\mathbf{A}_j}| : \mathbf{A}_j < 0\}$$

- When optimal but not feasible, apply simplex method but by finding leaving variable by choosing optimal element in $\hat{\mathbf{b}}$ and find entering variable by ratio test on $\hat{\mathbf{c}}^\intercal$. Ensure that
- When feasible but not optimal, apply regular simplex method.

Chapter 7

Sensitivity analysis on LPs

Often one may need to make slight adjustments to an optimization problem to reflect new conditions (cost of resource rises, amount of resources available depletes etc.). Sensitivity analysis studies how an optimization problem may be modified while retaining its feasible region, optimal solution or both. Here we discuss sensitivity analysis in the context of LPs; of which common modifications are the following.

- Small changes for entries of $\mathbf{c}$
- Small changes for entries of $\mathbf{b}$
- Small changes for entries of $\mathbf{A}$
- Appending a new constraint
- Appending a new variable

The main idea is that given an LP in standard form, we have the following conditions we can rely upon.

- $\hat{\mathbf{c}^\intercal} \leq \mathbf{0}$ is required to preserve optimality (for minimization problems).
- $\mathbf{B}^{-1}\mathbf{b} \geq \mathbf{0}$ is required to preserve feasibility.

Given that we have the $\mathbf{B}, \mathbf{N}$ of the optimal solution, we can create inequalities to determine how far entries can be changed while preserving optimality and feasibility.

7.1 Cost vector entries c

Small changes in $\mathbf{c}$ change the LP in different ways depending on whether the entry of $\mathbf{c}$ being changed is associated with a basis variable in the final tableau. Hence we will discuss the case of modifying nonbasis coefficients and basis coefficients separately.

7.1.1 Nonbasis

Denote changes in nonbasic cost vector entries as such.

$$\mathbf{c}_\mathbf{N}'^\intercal = \mathbf{c}_\mathbf{N}^\intercal + (\Delta\mathbf{c}_\mathbf{N})^\intercal$$

If the cost vector entry being modified is not related to a basic variable in the final BFS, we have the following bound.

$$(\Delta\mathbf{c}_\mathbf{N})^\intercal - \hat{\mathbf{c}_\mathbf{N}} \geq \mathbf{0} \implies \text{doesn't change optimal solution}$$

7.1.2 Basis

Denote changes in basic cost vector entries as such.

$$\mathbf{c}_\mathbf{B}'^\intercal = \mathbf{c}_\mathbf{B}^\intercal + (\Delta\mathbf{c}_\mathbf{B})^\intercal$$

If the cost vector entry being modified is related to a basic variable in the final BFS, we have the following bound.

$$(\Delta\mathbf{c}_\mathbf{B})^\intercal\mathbf{B}^{-1}\mathbf{N} + \hat{\mathbf{c}_\mathbf{N}}^\intercal \leq \mathbf{0} \implies \text{doesn't change optimal solution}$$

7.2 Constraint vector entries b

Since $\mathbf{b}$ has no bearing on the reduced cost vector, changes in $\mathbf{b}$ will have no effect on optimality, but could disrupt feasibility. Denote changes in the constraint vector as such.

$$\mathbf{b}' = \mathbf{b} + \Delta\mathbf{b}$$

Modifications to the constraint vector retain feasibility under the following condition.

$$\mathbf{B}^{-1}(\mathbf{b} + \Delta\mathbf{b}) \geq \mathbf{0} \implies \text{doesn't change feasible region}$$

7.3 Constraint matrix entries A

7.3.1 Nonbasis

$$\mathbf{A}'_j = \mathbf{A}_j + \Delta\mathbf{A}_j$$

$$\hat{\mathbf{c}}_j^\intercal + \mathbf{c}_\mathbf{B}^\intercal \mathbf{B}^{-1}(\Delta\mathbf{A}_j) \geq 0$$

7.3.2 Basis

The case where the entry is in a basis column is a bit more complex since our matrix $\mathbf{B}$ is being modified but we have $\mathbf{B}^{-1}$ on our formulae; we require a way to view how increments to a matrix entry modifies its inverse. A naiive approach is to calculate $\mathbf{B}$ from $\mathbf{B}$, and then calculate $(\mathbf{B} + \Delta\mathbf{B})^{-1}$. Then substitute this into the two inequalities we have been repeatedly using. Unfortunately, this approach is computationally expensive, however there exists a formula to assist in this situation.

The *Sherman-Morrison formula* can be employed to view how the inverse is effected by incremental arguments, and it can be used to derive the following.

7.4 Additional constraint

Check that optimal solution satisfies new constraint, if so optimal solution unchanged. if not, use row reductions to get in canonical form and add to tableau as a basis element. Note that it may be infeasible; in this case, continue using dual simplex method.

7.5 Additional variable

When appending a new variable to the LP, our main concern is whether the new variable's reduced cost entry of this variable retains optimality (i.e negative for min problems).

If (for a min problem) $\mathbf{c}_\mathbf{B}^\intercal \mathbf{B}^{-1}\mathbf{A}_j - \mathbf{c}_j \leq 0$, it can safely be entered into nonbasis without changing optimal solution, otherwise enter column $\mathbf{B}^{-1}\mathbf{A}_j$

into final tabealu and $\mathbf{c}_\mathbf{B}^\intercal \mathbf{B}^{-1}\mathbf{A}_j - \mathbf{c}_j$ into reduced cost and continue simplex method.

$\hat{\mathbf{c}_\mathbf{B}}^\intercal \mathbf{B}^{-1}\mathbf{A}_j - \mathbf{c}_j^\intercal \leq 0 \implies$ nonbasis entry doesn't change optimal solution

Part III

Nonlinear programming

Chapter 8

Unconstrained

For LPs, using the theory of convex analysis and linear algebra, we have the simplex method; an algorithm that derives *the* solution for that optimization problem. We can them preoccupy ourselves about reducing time complexity and understanding how 'continuously' changing the LP changes feasible regions and optimal solutions (sensitivity analysis).

A Nonlinear Program (NLP) is an optimization problem that is not an LP; maybe it has a nonlinear OF, or one, some or all constraints are nonlinear. Basically, such a problem is not an LP.

As one can imagine, this complicates matters imensely. LPs don't vary much in behaviour, but NLPs cover an extremely general range of problems; this variation between types of NLPs means that a 'one-size-fits-all' solution like the simplex method is not going to happen.

Real analysis, convex analysis, vector analysis, linear algebra and many other branches offer some critical theory in developing methods to solve NLPs, but alas, since generic functions are so diverse in behaviour we often resort to numerical methods to approximate the solution. Even so, this all assumes that the functions we're dealing with have some 'nice' properties like continuity and differentiability!

So we concede a little bit; we're going to conduct analysis on NLPs that are *unconstrained*, that is, their feasible regions are intervals, rectangles, entire spaces, and other types of nice convex sets.

- define NLPs as optimization problems as (x, f, g) Where $X = \{\mathbf{x} : \forall i \in \mathbb{N} \cap [1, n] a_i(\mathbf{x}) \leq \mathbf{0}\}$ Where at least one of the a_i or f isa nonlinear function. Actually, a formal definition for NLPs isn't really necessary; it's just an optimization problem that isn't an LP.

This chapter will look specifically at problems (X, f, g) Where $X = \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$ is a twice (totally) differentiable function

8.1 Theoretical basis

Like with LPs, we draw our theory from branches of mathematical analysis. For single variate real functions, the familiar derivative test provides the conditions when a differentiable function is at a local minimum.

Proposition 8.1 (Derivative test)**.** If $f'(x) = 0$ and $f''(x) > 0$, then x is a local minimum of f.

A similar test exists for multivariate real functions.

Recall the positive definite property of a matrix; note that we can verify that a matrix is positive definite by assuring that all eigenvalues are positive.

Proposition 8.2 (Multivariate derivative test)**.** If a function f with continuous second derivatives has $\nabla f(\mathbf{x}_0) = \mathbf{0}$ and $\mathbf{H}_f(\mathbf{x}_0)$ is positive definite, then x_0 is a local minimum of f.

Some results from convex analysis can help us deduce which local minima are also global minima.

Proposition 8.3. If f is convex and $\mathrm{dom}(f)$ is a convex set, any local minimum of f is a global minimum.

8.2 Derivative tests

Proposition 8.4 (Derivative test)**.** If a real function f with continuous second derivatives has $f'(x_0) = 0$ and $f''(x_0) > 0$, then x_0 is a local minimum of f.

Proposition 8.5 (Multivariate derivative test)**.** If a real multivariate function f with continuous second derivatives has $\nabla f(\mathbf{x}_0) = \mathbf{0}$ and $\mathbf{H}_f(\mathbf{x}_0)$ is positive definite, then x_0 is a local minimum of f.

Proposition 8.6. Let f have continuous second-order partial derivatives on U. Then f is convex on U iff its Hessian matrix $\mathbf{H}_f$ is positive semi-definite.

Proposition 8.7. Let f have continuous second-order partial derivatives on U. If the Hessian matrix $\mathbf{H}_f$ is positive definite on U, then f is convex on U.

8.3 Gradient methods

We now have sufficient theory to develop the following numerical methods that are applicable to many differentiable real functions.

8.3.1 Steepest descent method

- ε is the tolerance
- f is the multivariate differentiable function to be minimized

---

while $\|\nabla f(\mathbf{x})\| < \varepsilon$ **do**
  $\mathbf{d} \leftarrow -\nabla f(\mathbf{x})$
  $t_0 \leftarrow \operatorname{argmin}_{t>0} f(\mathbf{x} + t\mathbf{d})$
  $\mathbf{x} \leftarrow \mathbf{x} + t_0\mathbf{d}$
end while

8.3.2 Newton's method

- ε is the tolerance
- f is the multivariate differentiable function to be minimized

---

while $\|\nabla f(\mathbf{x})\| < \varepsilon$ **do**
  $\mathbf{x} \leftarrow \mathbf{x} - \mathbf{H}(f(\mathbf{x}))^{-1}(\nabla f(\mathbf{x}))$
end while

Chapter 9

Constrained

This chapter will instead stuy problems of the form (X, f, g), where $f : \mathbb{R}^n \to \mathbb{R}$ is a twice (totally) differentiable function, although $X = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} = \mathbf{b}\}$

9.1 Reduced function

These types of problems can be converted to unconstrained NLPs.

9.1.1 Single linear condition

9.1.2 Generalization to multiple linear conditions

$$\mathbf{x} = \mathbf{x_b} + \mathbf{x_N}$$

$$\mathbf{x} = \mathbf{B}^{-1}(\mathbf{x_B} + \mathbf{x_N})$$

$$\mathbf{x} = \overline{\mathbf{x}} + \mathbf{Z}\mathbf{x_N}$$

$$\phi(\mathbf{x_N}) = f(\overline{\mathbf{x}} + \mathbf{Z}\mathbf{x_N})$$

This is the *reduced function of* f, notice that $\mathbf{Z}$ is a basis for the kernel of $\mathbf{A}$. We realize that the chain rule permits a simplified form of the gradient and Hessian of the reduced function in terms of the original function.

$$\nabla\phi(\mathbf{x_N}) = \mathbf{Z}^\intercal \nabla f(\mathbf{x})$$

$$\mathbf{H}(\phi(\mathbf{x_N})) = \mathbf{Z}^\intercal \mathbf{H}(f(\mathbf{x}))\mathbf{Z}$$

Using these, so long as we can calulate $\mathbf{Z}$, our constrained NLP becomes an unconstrained NLP; from here we can apply numerical methods and derivative tests.

9.2 Lagrangian relaxation method

$$\mathcal{L}_f(\mathbf{x}, \Lambda) = f(\mathbf{x}) + \sum_{i=1}^{m} \Lambda_i (b_i - g_i(\mathbf{x}))$$

One can prove that optimizing the Lagrangian $\mathcal{L}_f$ also optimizes f, hence we have again mapped a constrained NLP to an unconstrained NLP.

Since stationary points occur when $\nabla \mathcal{L}_f(\mathbf{x}) = \mathbf{0}$, stationary points of the Lagrangian occur on the following condition.

$$\nabla f(\mathbf{x} = -\sum_{i=1}^{m} \Lambda_i \nabla g_i(\mathbf{x})$$

The following proposition helps us determine the nature of stationary points when using the Lagrangian on a convex function.

Proposition 9.1. If f is convex, then the optimal values of $\mathcal{L}_f$ are minima.

Part IV

Integer programming

Chapter 10

Integer programs

Some optimization problems require solutions to be integers; such an optimization problem is called an *integer program (IP)*. There are different types of IPs that one may encounter.

- Pure IP; All variables are restricted to $\mathbb{Z}$
- Binary IP; All variables are restricted to $\{0, 1\}$
- Mixed IP; Some ariables are restricted to $\mathbb{Z}$

This part will only consider Linear integer programs for simplicity.

Chapter 11

Branch-and-bound method

11.1 Main idea

A naive approach of a 'first approximation' is to relax any integer number requirements to real number requirements and solve the relevant LP. Imagine the index $\mathbf{x}_j$ of the 'optimal solution' is not an integer, we *branch* into two 'subproblems' with additional constraints $\mathbf{x}_j \leq \lfloor \mathbf{x}_j \rfloor$ and $\mathbf{x}_j \geq \lfloor \mathbf{x}_j \rfloor + 1$ respectively.

When a subproblem yields an 'integer optimal solution', it i called a candidate solution, since it has potential to be the overall solution. THis subproblem can't be improved further, so we call it 'fathomed'. We also fathom subproblems that are infeasible or unbounded.

The best candidate solution is called the 'incumbent solution', and the incumbent solution once all subproblems have been fathomed is the optimal solution.

With candidate solutions, for min problems, our optimal value is *bounded* by the optimal value of the current incumbent solution's optimal value. Therefore, if a subproblem produces a result above this value, it fails to replace the incumbent solution. If it is less than this bound and it is an integer solution, it becomes the new incumbent solution.

11.2 Algorithm

Chapter 12

Hungarian method

Assignment problem

$$X = \{(\mathbf{X} : m \times m) : \sum_{j=1}^{m} X_{ij} \wedge \sum_{i=1}^{m} X_{ij} \wedge \mathbf{X}_{ij} \in \{0, 1\}\}$$

$$f(\mathbf{X}) = \sum_{i=1}^{m} \sum_{j=1}^{m} \mathbf{C}_{ij} \mathbf{X}_{ij}$$

It gets its name due to the amount of Hungarian mathematicians working on the problem (the martians)

Chapter 13

Cutting plane method

pick row in final tableau with RHS not an integer Add 'the cut' of this row as a new constraint, continue with dual simplex method Repeat until integer solution obtained