Lecture Notes*:
MATH 323 – Operations Research: Deterministic Models

Daniela Hurtado-Lange
William & Mary

Spring 2023

*This note is based on: Optimization in Operations Research, by Ronald Rardin. Pearson (2017).

Contents

1 Introduction

In this course we will learn how to solve and analyze **operations** problems such as planning work shifts, choosing investments, demand satisfaction at minimum cost, designing facilities, finding the shortest path between two locations, and many more. More generally, the type of problems we want to analyze involve a **decision making** process where we want to find the **best way** to operate a system while **satisfying key properties** defined by each situation.

In many cases, there are several actions we can take to make things "better", but they may involve a cost. In other words, we need to find the best way to trade off our actions. Let's start with the following example.

Example 1.1. *[Application 1.1 from the textbook] Mortimer Middleman (MM) operates a wholesale diamond business. He needs to travel to Belgium to buy the diamond supply at the international market. Each carat of diamond costs \$700, and he is required to buy at least 100 carats each trip. Each trip to Belgium takes 1 week and costs \$2000.*

The selling price per carat is \$900. Historical data shows that MM has variable demand and, hence, sometimes he runs out of diamonds. His customers are not very patient, so they will not pre-order diamond. In other words, MM must have diamonds in stock whenever a customer comes, or the customer goes away.

On the other hand, MM feels that holding too much stock is not good for his business because his insurance cost increases. Additionally, having too much stock ties up capital that he could otherwise be investing.

What is the best way to operate the business?

Solution. There are several things to consider before we can find the best way to operate the business. For example:

- The ultimate goal of MM is to maximize profit and minimize costs
- MM makes profit by selling carats of diamonds
- If a customer comes to the store and there are no diamonds, we lose the customer (no profit). Hence, we always want to keep some carats in stock.
- Following the same logic, MM should restock as soon as his stock start getting low.
- Restocking has a cost because MM needs to travel, and traveling costs money. Hence, we do not want MM to travel too often.
- Every time MM goes to Belgium, he needs to buy at least 100 carats. This does not bring any issue in principle. However, there is a holding cost (as described later).
- There is a holding cost for keeping diamonds in the store, so we do not want to keep too many carats.
- The demand may be uncertain (we only know its mean)
- etc...what else can you come up with?

□

1.1 Formalizing the discussion

In small problems, we may be able to find the best solution by trial and error. However, this approach can be very expensive in terms of time and money. For example, if MM uses this approach, they may hold a lot of stock some times, and run out of diamonds a few times before they learn when they should replenish their stock. There is a direct cost in money and time here, but we may additionally consider some indirect costs. For example, if a particular customer comes to the store when there are no diamonds, they may never come back.

In this course, we will learn how to mathematically formulate real-life decision problems, such as MM's problem, and then we will solve them using the software AMPL. We will start learning to model problems, that is, some conventions that will help us quantify the relationships needed to represent each real-life situation, the goal(s) we pursue, and the key features that make our situation unique.

Let's start with some key concepts.

Definition 1.1. *[Def. 1.2 from the textbook] The three fundamental concepts in operations research models are:*

(a) Decisions open to decision makers

(b) Objectives making some decisions preferred to others

(c) Constraints limiting the decision choices

To find the decisions, constraints and objectives in each case, we can ask ourselves the following questions:

(a) Decisions: How can we make things better? Or, more specifically, what are the variables we can *move* to make things better?

(b) Objectives: What makes one decision better than another?

(c) Constraints: What prevents us from doing whatever we want? What features make this problem unique?

Example 1.2. *Let's apply the definition to MM's company. MM is the decision maker and his:*

*(a) Decisions: **when** to reorder and **how many** carats to purchase each time.*

(b) Objective: minimize the operational cost (traveling and holding cost) of his company. Equivalently, we can say that his objective is to maximize profit, where profit = revenue - cost.

(c) Constraints: buy at least 100 carats each time, and inventory needs to be nonnegative. A typical constraint is also the storage room, that is, the physical space he has to store diamonds.

In the rest of this course, we will build optimization models that represent operational problems, and we will learn algorithms to solve them. Let's start with the definition of an optimization model.

Definition 1.2 (Def. 1.3 from the textbook)**.** *Optimization models represent problem choices as decision variables and seek values that maximize or minimize objective functions of the decision variables subject to constraints on variable values expressing the limits on possible decision choices.*

1.2 More examples

In Example 1.1 we studied a company that needs to maximize profit by smartly restocking their product. Let's see some other examples where the objective and variables are a bit different. You may find some additional examples in Chapter 1 of the textbook.

For each of the following cases, identify the decisions, constraints and objectives.

Example 1.3 (Traveling Salesperson Problem, aka, TSP)**.** *MM wants to expand his business, and he starts offering delivery within a radius of 25 miles. Suppose that the delivery area can be split into 10 zones and all the orders from each zone can be instantaneously delivered at the same address. Customers can place online orders 24/7, and MM promises delivery on the following Friday. MM is not very good with directions, so he wants to follow the same route every single Friday. Hence, he needs to find a route that visits all the zones. Additionally, he starts and ends at the store.*

How should MM plan his route to minimize cost?

Solution. Just as in Example 1.1, the decision maker is MM. However, now he's solving a different problem. His decisions, constraints and objectives are:

(a) Decisions: In which order to visit the zones

(b) Objective: Minimize delivery cost or, equivalently, find the shortest path

(c) Constraints: Start and end at the store, and visit all the zones.

□

The above example represents a typical TSP, where the goal is to find the minimum-cost path while creating a route that visits all locations at least once.

Example 1.4 (Knapsack problem)**.** *As a student who is taking an optimization class, you decide to use what you are learning to organize your schedule. You have the following requirements:*

(i) You want to sleep at least 8 hours every day.

(ii) You are taking 4 classes this semester and you estimate that each class will need 12 hours each week, considering the lectures and the extra work you need to do.

(iii) After all these years being a student, you know yourself, and you have realized that you need to rest 30 minutes every two hours of work to maintain your productivity.

(iv) Your 2022 resolution was to be healthier. Then, you want to go to the gym every day for 1 hour, and you want to eat a homemade dinner every day. After going to the gym, you take a shower and eat a (healthy) snack. These take 1 hour in total. Hence, going to the gym takes 2 hours of your day.

(v) To keep it realistic with dinner, you decide to cook three times a week and you cook enough food for 2-3 days. Every time you cook, it takes you 3 hours considering the actual cooking process and doing the dishes.

(vi) To eat healthier you also need some time to go grocery shopping every week. Suppose you take 2 hours to do that.

(vii) Part of having a healthier life is to take enough breaks during the week. Then, you want to take 6 hours every Sunday for yourself. You may go out with friends, sleep all day, go to the beach, etc. Whatever you do, you will not do work, cooking or gym.

(viii) After a tiring day with cooking and the gym, you need to rest two hours before continuing with your life. Then, whenever you cook you take additional 2 hours of rest, and when you don't cook you take 1.

(ix) You do want to be healthy, but you don't want to lose your essence. Something you love and you don't want to give up because of a new year's resolution, is your time at Aromas. You go three times a week and you spend two hours (considering the time it takes you to get there, order and have your coffee).

(x) You estimate that other daily activities, such as taking a shower, having your meals, transportation, etc takes 5 hours of your day.

A week only has 168 hours, so the first thing you realize when you look at your requirements, is that you won't be able to do everything. What subset of activities should you choose?

Solution. In this problem, you are the decision maker. Your decisions, constraints and objective are:

(a) Decisions: What activities to do every week

(b) Objective: Maximize your happiness. Actually, to solve this problem you should decide how much you value each of the activities described in the problem.

(c) Constraints: The amount of time you use in your activities should not exceed 168 hours (total number of hours in one week), the relationship between the activities and breaks.

□

The example above is a variation of a knapsack problem. In a knapsack problem, you have a bag with finite volume and you want to maximize the value of the items you can fit inside. Each item has a positive volume, and the total volume of all the items exceeds the capacity of the knapsack. Then, how should we choose which items we take and which ones we leave? In our example, the items are activities and the volume of the bag is the amount of time available in one week.

Example 1.5 (Scheduling)**.** *A hospital needs to plan the shifts of the nurses in a specific unit. There are 10 available nurses. Depending on their experience, each nurse is expert in 3 main areas. In the hospital unit, there are 5 skills that are needed at every time. Then, at every time, at least one of the nurses on duty needs to be an expert on each of these 5 areas.*

Due to the number of beds in the unit, day shifts require 3 nurses on duty and night shifts, 2. After completing a day shift, a nurse needs to rest for at least one shift, and after completing a night shift, they need to rest for 4 shifts. If we know their expertise, what is the best way to schedule the shifts?

Solution. In this case, the decision maker is the hospital management. Their decisions, constraints and objective are:

(a) Decisions: Which nurses to call for each shift

(b) Objective: In this case the objective is to find a feasible schedule. If there are two feasible schedules, we currently don't have enough information to decide which one we prefer, so any of them will be okay. We can add more information to the problem, such as, preferences of the nurses in terms of day/night shift, or teams that work better than others.

(c) Constraints: Each team of nurses needs to have the 5 required skills, and they need to rest after each shift. Also, there are only 10 available nurses

Another option would be to allow breaking the rule of number of shifts they need to rest, and add a cost for breaking it. Then, the objective would be to minimize the cost.

□

2 Deterministic Optimization Models in Operations Research

[This section is based on Chapter 2 of the textbook]

This is one of the most important chapters of this course. An essential skill in Operations Research is to construct mathematical models that represent the decision-making processes we face. In this chapter, we will learn several techniques to represent some of the most frequently used models and we will work together on several examples.

2.1 Decision variables, constraints and objective functions

Recall in Definition 1.1 we discussed decisions, constraints and objectives in words, and later we assigned a question to each concept. In this section, we will formalize these definitions to associate a mathematical formulation to the decision-making problem.

We start with the definitions, and then we go through some examples. The first concept we will define is decision variables, which (as their name is saying) represent the decisions we can make.

Definition 2.1. *[Def. 2.1 from textbook] Decision variables are the variables in the optimization problem that represent the decisions to be taken.*

There are usually many factors playing in Operations Research problems, and we might feel overwhelmed by the amount of information. The first step is to distinguish input parameters (or data) from variables. The input parameters are the information that is given, and we cannot do anything to change them. The decision variables, on the other hand, are related to the decision we make to operate the system more efficiently. In other words, **the decision maker gives us the input parameters, and we find the best value for the decision variables** (once we solve the problem, of course).

We use the decision variables to construct the optimization model, by writing the objective and the constraints mathematically.

Definition 2.2 (Def. 2.4 from the textbook)**.** *Objective functions in optimization models quantify the decision consequences to be maximized or minimized.*

In other words, the objective function provides a value to the decisions and, hence, it allows us to compare options and decide which one is better.

We now define the constraints, which limit the values of the decision variables so that the problem makes sense.

Definition 2.3. *[Def. 2.2 and 2.3 from textbook]*

(a) Variable-type constraints specify the domain of definition for the decision variables. In other words, they specify the set of values for which the variables have meaning.

(b) Main constraints specify the restrictions and interactions, other than the variable type, that limit the decision variable values.

The variable-type constraints are simple, yet very important. Failure to write them in an optimization problem can lead to meaningless solutions because we are trying to develop a model that will be solved by a software. Hence, we cannot assume that the software will have some "common sense" and avoid certain types of solutions. **We need to tell the software what is an acceptable solution**.

Some of the most popular variable types are: real numbers (also known as unrestricted), nonnegative real numbers, integers, nonnegative integers and binary.

Let's see an example.

Example 2.1. *[Application 2.1 from the textbook] Two Crude Petroleum runs a small refinery on the Texas coast. The refinery distills crude petroleum from two sources, Saudi Arabia and Venezuela, into three main products: gasoline, jet fuel and lubricants.*

The two crudes differ in chemical composition and thus yield different product mixes. Each barrel of Saudi crude yields 0.3 barrel of gasoline, 0.4 barrel of jet fuel, and 0.2 barrel of lubricants. On the other hand, each barrel of Venezuelan crude yields 0.4 barrel of gasoline, 0.2 barrel of jet fuel and 0.3 barrel of lubricants. The remaining 10% of each barrel is lost to refining.

The crudes also differ in cost and availability. Two Crude can purchase up to 9000 barrels per day from Saudi Arabia at $100 per barrel. Up to 6000 barrels per day of Venezuelan petroleum are also available at the lower cost of $75 per barrel because of the shorter transportation distance.

Two Crude's contract with independent distributors require it to produce 2000 barrels per day of gasoline, 1500 barrels per day of jet fuel and 500 barrels per day of lubricants. How can these requirements be fulfilled most efficiently?

Identify the decision variables, constraints and objective function.

Solution. The **decision variables** are the barrels of crude that we order from Saudi Arabia and Venezuela in one day. When we formulate an optimization problem, we must define the decision variables and assign a symbol to each of them. In this case, let

$$x_1 \triangleq \#\text{ barrels of Saudi crude refined per day (in thousands)}$$
$$x_2 \triangleq \#\text{ barrels of Venezuelan crude refined per day (in thousands)}$$

The **objective** is to operate the system at minimum cost. The only costs we know of are related to buying crude. Since the cost of each barrel of Saudi crude is $100 and of Venezuelan crude is $75, we obtain the following objective function:

$$\min \quad 100x_1 + 75x_2.$$

Both variables are nonnegative real numbers since we cannot refine a negative number of barrels, but we may refine 1/2 barrel. Hence, the **variable-type constraints** are:

$$x_1 \geq 0, \quad x_2 \geq 0.$$

The **main constraints** describe how the variables interact with each other. In this case, we have two sets of main constraints, that are explained below:

- Product requirement by contract with buyers: There is a minimum number of barrels of final product (gasoline, jet fuel and lubricants) that Two Crude Petroleum needs to produce every day. Then, we must have

  $$\begin{aligned} &\text{Total gasoline produced} \geq 2000, \\ &\text{Total jet fuel produced} \geq 1500, \quad \text{and} \\ &\text{Total lubricants produced} \geq 500 \end{aligned}$$

  To compute the total amount produced of each product, we consider the % of crude that can be transformed into each product depending on their origin. For example, 30% of the Saudi crude and 40% of the Venezuelan crude becomes gasoline. Hence, the total amount of gasoline produced is $0.3x_1 + 0.4x_2$. Following a similar approach for the jet fuel and lubricants, and remembering that x_1 and x_2 are measured in thousands of barrels, we obtain

  $$\begin{aligned} 0.3x_1 + 0.4x_2 &\geq 2 \qquad \text{(gasoline need)} \\ 0.4x_1 + 0.2x_2 &\geq 1.5 \qquad \text{(jet fuel need)} \\ 0.2x_1 + 0.3x_2 &\geq 0.5 \qquad \text{(lubricant need)} \end{aligned}$$

- Availability of crude from each source: There is a maximum amount of crude that we are allowed to buy from each source. Then, our decision variables are upper bounded. We obtain

  $$\begin{aligned} x_1 &\leq 9 \qquad \text{(Saudi crude)} \\ x_2 &\leq 6 \qquad \text{(Venezuelan crude).} \end{aligned}$$

□

Now that we have the decision variables, constraints and objective, we want to write the entire optimization problem. We use the following format.

Principle 2.1 (Principle 2.1 from the textbook)**.** *The standard form of an optimization model has the form:*

$$\begin{aligned} \min \text{ or } \max \quad & \textit{Objective function} \\ \textit{s.t.} \quad & \textit{Main constraints} \\ & \textit{Variable-type constraints}, \end{aligned}$$

where "s.t." stands for "subject to".

Hence, we may write the Two Crude Refinery optimization problem as follows:

$$\begin{aligned} \min \quad & 100x_1 + 75x_2 \\ \text{s.t.} \quad & 0.3x_1 + 0.4x_2 \geq 2 \qquad \text{(gasoline need)} \\ & 0.4x_1 + 0.2x_2 \geq 1.5 \qquad \text{(jet fuel need)} \\ & 0.2x_1 + 0.3x_2 \geq 0.5 \qquad \text{(lubricant need)} \\ & x_1 \leq 9 \qquad \text{(Saudi crude availability)} \\ & x_2 \leq 6 \qquad \text{(Venezuelan crude availability)} \\ & x_1, x_2 \geq 0 \end{aligned}$$

Let's do another example.

Example 2.2 (Example 2.1 from the textbook)**.** *Suppose that we have 80 meters of fence and we want to enclose a rectangular yard. Formulate an optimization model to find the design of maximum area.*

Solution. We first identify the decision variables. A rectangle is characterized by its length and width, and these are exactly our decision variables. Let

$$\begin{aligned} \ell = & \text{ Length of the yard} \\ w = & \text{ Width of the yard} \end{aligned}$$

The objective is to maximize the area of the yard. Then, the objective function is:

$$\max \quad \ell w.$$

Both variables are nonnegative numbers. Then, the variable-type constraints are:

$$\ell,\, w \geq 0$$

The only constraint to our problem is that we have 80 meters of fence and, therefore, the perimeter of the rectangle cannot be larger than that. Then, our main constraint is

$$2\ell + 2w \leq 80.$$

Therefore, our optimization problem is:

$$\begin{aligned} \max \quad & lw \\ \text{s.t.} \quad & 2\ell + 2w \leq 80 \\ & \ell, w \geq 0. \end{aligned}$$

□

Observe that, since we only have 80 meters of fence, both decision variables, ℓ and w, should be upper bounded by 80. Then, **why didn't we include $\ell \leq 80$ and $w \leq 80$ in our main constraints?** Observe that if the constraint $2\ell + 2w \leq 80$ is satisfied, we trivially have that $\ell \leq 80$ and $w \leq 80$. Then, we may include $\ell \leq 80$ and $w \leq 80$ in the main constraints, but they would be redundant constraints. In small problems, as the examples above, having redundant constraints won't make any difference. However, if we are dealing with large problems, we want to avoid any redundancy to reduce the time the software takes to solve the problem.

2.2 Graphic solution and optimization outcomes

Before we move on to modeling larger optimization problems, we will study the simplest solution method. This method is not frequently used in practice, but it will help us build the intuition we need to better understand other solution methods. In this section, we will solve optimization problems **graphically** and we will use the drawings to analyze some of the most common issues we may face for larger problems.

We start with some definitions.

Definition 2.4. *Consider an optimization model.*

(a) (Def. 2.6 of the textbook) The feasible set is the collection of choices for decision variables satisfying all the model constraints.

(b) A feasible solution is a choice of the decision variables that belongs to the feasible set.

(c) (Def. 2.18 of the textbook) The optimization model is said to be infeasible if no choice of decision variables satisfies all the constraints. In other words, if the optimization problem does not have any feasible solution.

(d) (Def. 2.12 of the textbook) In a minimization/maximization problem, an optimal solution is a feasible choice for decision variables where the objective function value is smaller/greater than or equal to any other feasible solution.

(e) (Def. 2.14 of the textbook) The optimal value in an optimization model is the objective function value of any optimal solution.

Observe that when we defined optimal solutions, we used the article "an" and when we defined optimal value, we used "the". This is a hint to the idea that we may have multiple optimal solutions, but there is only one optimal value. We will see some examples once we learn the graphical method.

In the graphical method, we plot the feasible region and curves representing the objective function to find optimal solutions. Hence, we can only use it for problems that have 2 or 3 decision variables. Specifically, we follow the steps described below. To simplify the notation, let's suppose we have two decision variables and call them x_1 and x_2. The same steps apply if we have three decision variables.

Algorithm 2.1 (Graphical method)**.** *We split the method in three main steps, that we describe below:*

1. *Draw the feasible region:*
   - 1.1. *Assign one coordinate axis to each decision variable*
   - 1.2. *Draw the variable-type constraints*
   - 1.3. *For each of the main constraints:*
     - i. *If the constraint is an inequality, replace the inequality by an equality*
     - ii. *Draw the curve (usually a line in this course) described by the equality constraint*
     - iii. *If the original constraint is an inequality, identify which side of the curve you drew satisfies the inequality. To do this, it is sufficient to try one point (you may use the origin). If the point satisfies the inequality, you are on the correct side of the curve. If not, pick the other side.*
2. *Draw curves showing the objective function value and detect the decreasing/increasing direction: Let's denote the objective function by $c(x_1, x_2)$, and observe this is a three-dimensional curve.*
   - 2.1. *Choose a feasible solution (x_1', x_2') and evaluate the objective function. Call $y' = c(x_1', x_2')$*
   - 2.2. *Draw the curve $y' = c(x_1, x_2)$. Observe that now we fixed the objective function value y' and we are taking x_1 and x_2 as variables again.*
   - 2.3. *Compute the gradient of the objective function:*

$$\nabla c(x_1, x_2) \triangleq \left(\frac{\partial}{\partial x_1} c(x_1, x_2),\ \frac{\partial}{\partial x_2} c(x_1, x_2) \right).$$

$\nabla c(x_1, x_2)$ *is the increasing direction of the objective function, and* $-\nabla c(x_1, x_2)$ *is the decreasing direction.*

2.4. *For a minimization/maximization problem, find another solution that is in the decreasing/increasing direction starting from the point* (x_1', x_2') *that you chose before.*

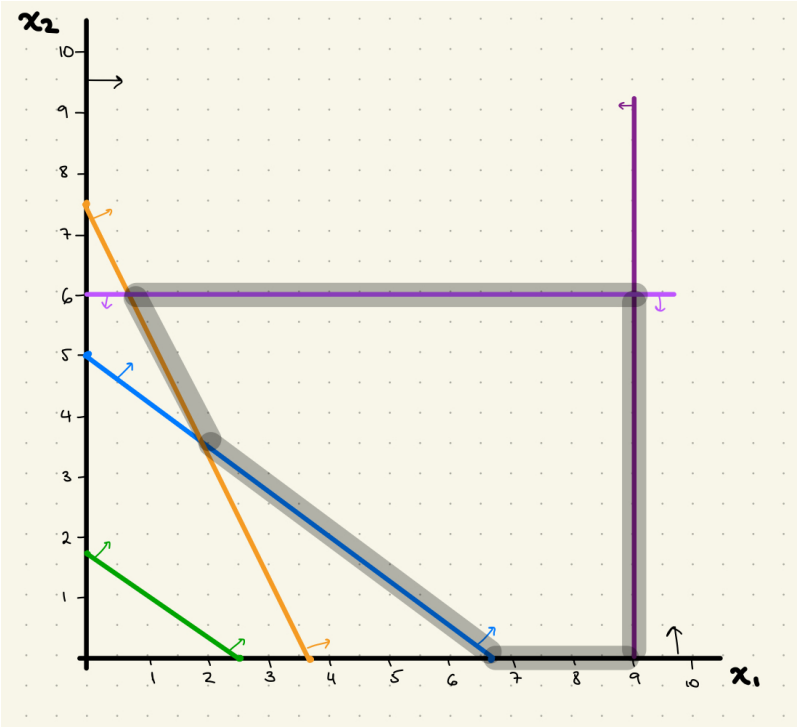
3. *Find optimal solutions and the optimal value:*
You will know you are at an optimal solution if moving away from the current point makes the objective function value worse. Specifically, in a minimization/maximization problem if:

- *You move in the decreasing/increasing direction, and you end up in an infeasible point (i.e., outside the feasible set)*
OR
- *You moved in the decreasing/increasing direction before, and wherever you move now, your objective function gets larger/smaller or equal. In other words, you moved in the improving direction in the previous step, but no matter where you move now, your solution does not improve.*

Let's illustrate the graphical method with some examples. We start solving the Two Crude Petroleum optimization problem. We had the following optimization problem:

$$\begin{aligned} \min \quad & 100x_1 + 75x_2 \\ \text{s.t.} \quad & 0.3x_1 + 0.4x_2 \geq 2 \\ & 0.4x_1 + 0.2x_2 \geq 1.5 \\ & 0.2x_1 + 0.3x_2 \geq 0.5 \\ & x_1 \leq 9 \\ & x_2 \leq 6 \\ & x_1, x_2 \geq 0. \end{aligned}$$

Then, matching the color of the constraints with the lines, we obtain that the feasible region is the area marked on the thick gray lines in the following picture:



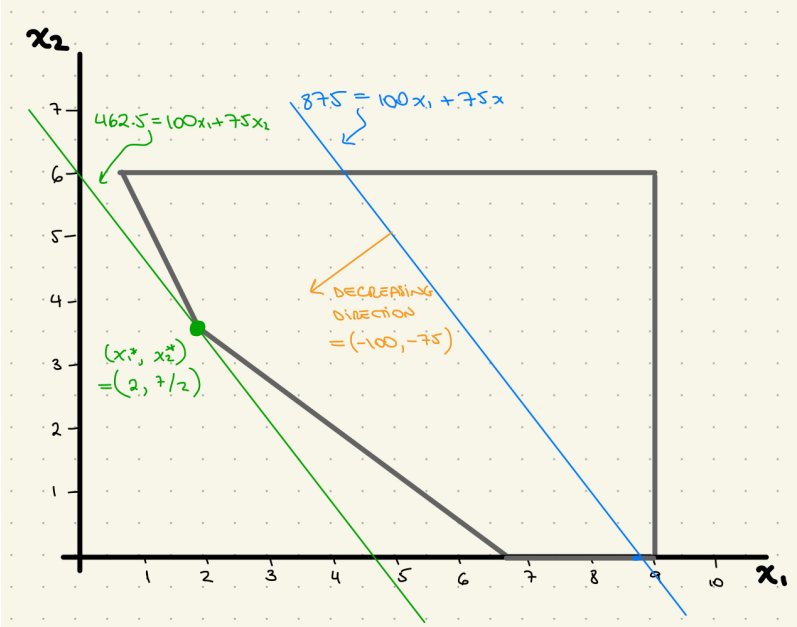
This completes step 1. from the graphical method algorithm. To complete step 2, we define $c(x_1, x_2) = 100x_1 + 75x_2$ and do:

2.1. Choose a feasible solution $(x_1', x_2') = (5, 5)$ and realize that $y' = c(5, 5) = 875$

2.2. Draw the curve $875 = 100x_1 + 75x_2$

2.3. Compute the gradient of the objective function and identify the decreasing direction $-\nabla c(x_1, x_2) = (-100, -75)$

Then, moving in the decreasing direction, we find that the optimal solution is $(x_1^*, x_2^*) = (2, 3.5)$ with optimal value 462.5. Observe that if we move a little bit more in the decreasing direction, we leave the feasible region. We plot this process in the following picture:



The example above is, in some sense, ideal. We had a bounded feasible region, and there was a unique optimal solution. However, this is not always the case. There are some optimization problems that have multiple optimal solutions, where an optimal solution does not exist, or even where the set of constraints lead to an infeasible problem. We exemplify these cases in the examples below.

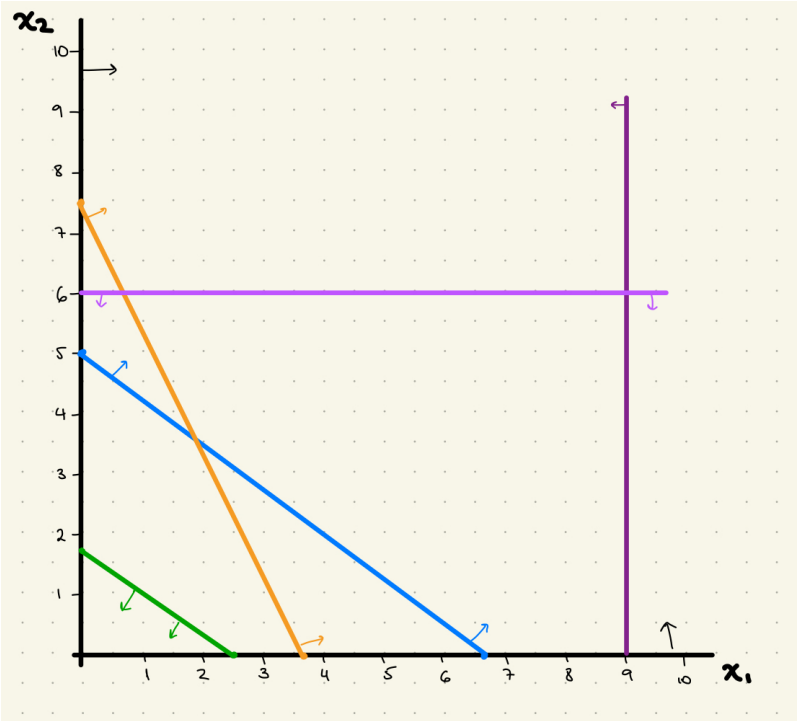
Example 2.3. *Consider the following optimization problem. Use the graphical method to solve it.*

$$\begin{aligned} \min \quad & 100x_1 + 75x_2 \\ s.t. \quad & 0.3x_1 + 0.4x_2 \geq 2 \\ & 0.4x_1 + 0.2x_2 \geq 1.5 \\ & 0.2x_1 + 0.3x_2 \leq 0.5 \\ & x_1 \leq 9 \\ & x_2 \leq 6 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Solution. Observe that it is almost the same problem as the Two Crude Petroleum problem. The only difference is the direction of the inequality on the third constraint. Let's first draw the feasible region, using the following color code:

$$\begin{aligned} \min \quad & 100x_1 + 75x_2 \\ \text{s.t.} \quad & 0.3x_1 + 0.4x_2 \geq 2 \\ & 0.4x_1 + 0.2x_2 \geq 1.5 \\ & 0.2x_1 + 0.3x_2 \leq 0.5 \end{aligned}$$

$$x_1 \leq 9$$
$$x_2 \leq 6$$
$$x_1, x_2 \geq 0.$$



Observe that the intersection of all the inequalities is the empty set. In other words, there is no point (x_1, x_2) that satisfies all the constraints. □

Now let's see an example where, only changing the objective function, we observe a unique solution, multiple solutions, and no solution due to unboundedness.

Example 2.4. *Consider the set*

$$\mathcal{P} = \{(x_1, x_2) \in \mathbb{R} : \ 0.3x_1 + 0.4x_2 \geq 2, \ 0.4x_1 + 0.2x_2 \geq 1.5, \ 0.2x_1 + 0.3x_2 \geq 0.5, \ x_1, x_2 \geq 0\}.$$

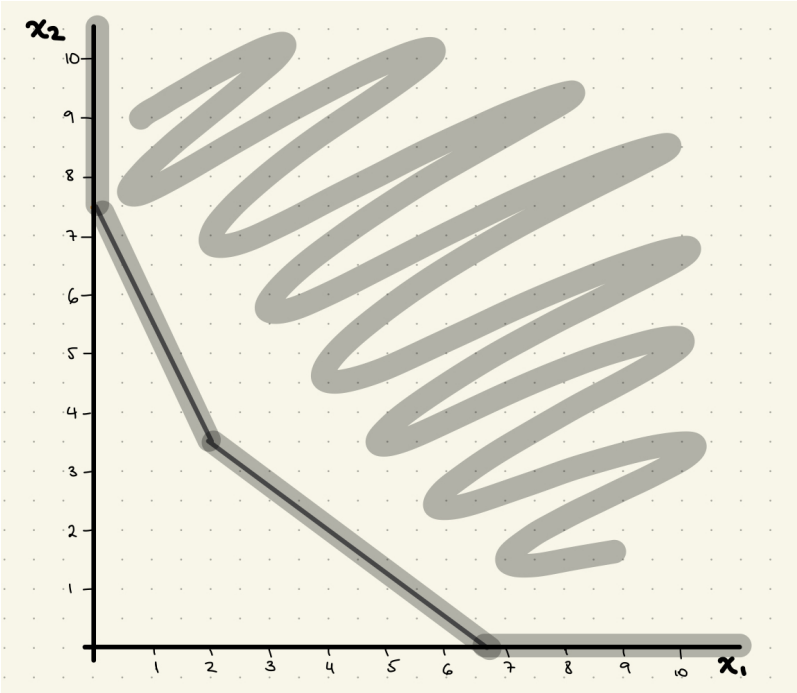
Solve the following optimization problems:

(a) $\min\{100x_1 + 75x_2 : \ (x_1, x_2) \in \mathcal{P}\}$

(b) $\min\{90x_1 + 120x_2 : \ (x_1, x_2) \in \mathcal{P}\}$

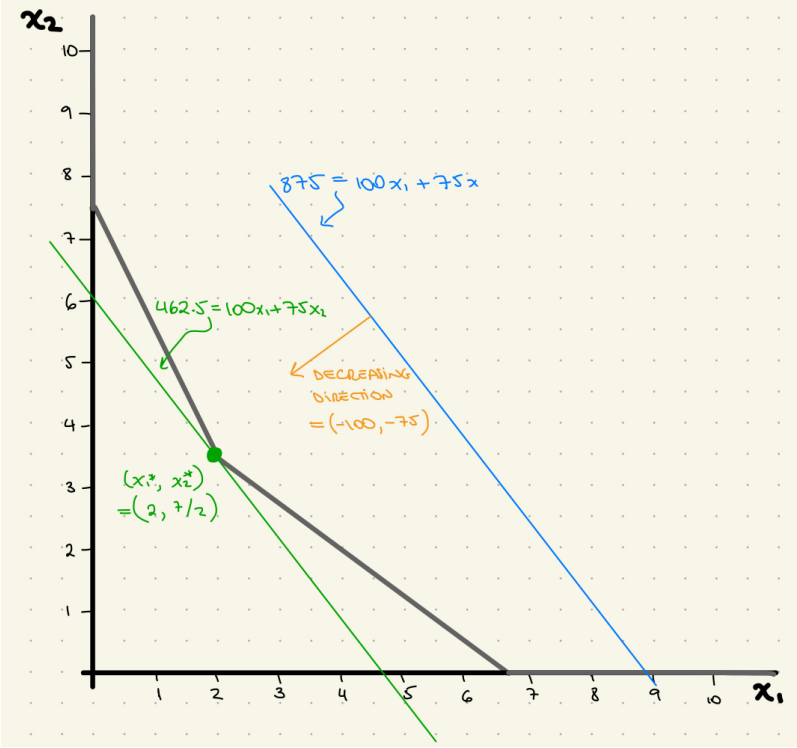
(c) $\max\{100x_1 + 75x_2 : \ (x_1, x_2) \in \mathcal{P}\}$

Solution. First observe that the set $\mathcal{P}$ is almost equal to the feasible set of the original Two Crude Petroleum problem. The only difference is that we got rid of the upper bound of the decision variables. Then, the set $\mathcal{P}$ is the following:



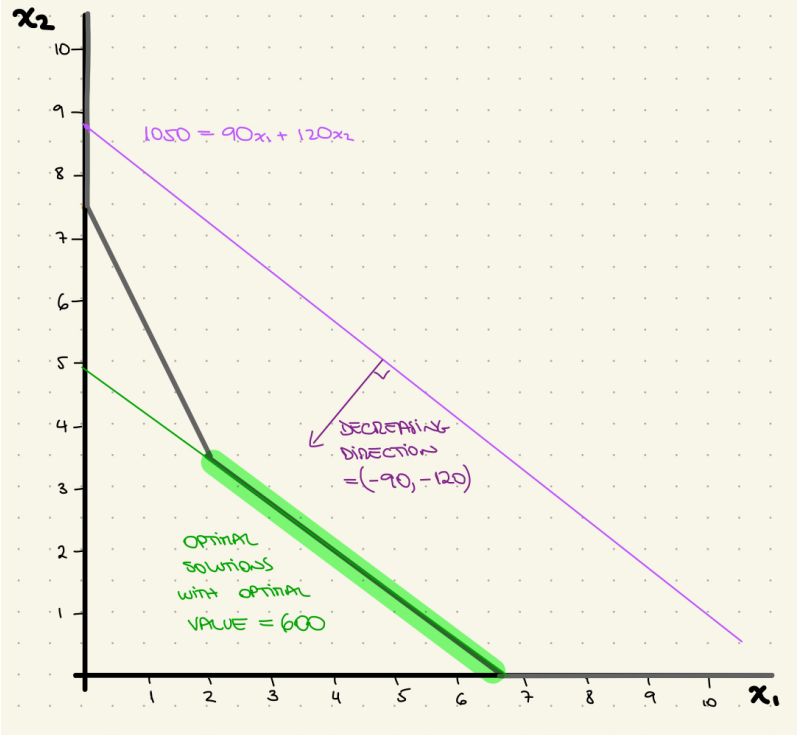
In the three problems, we present an alternative way of writing an optimization problem. Here, instead of listing all the constraints, we define the feasible set and use it to write the optimization problem more concisely. Let's solve each of them graphically.

(a) Observe that the objective function of this problem is exactly the same as in the Two Crude Petroleum problem. Following the same steps as before, we observe that we obtained exactly the same optimal solution.



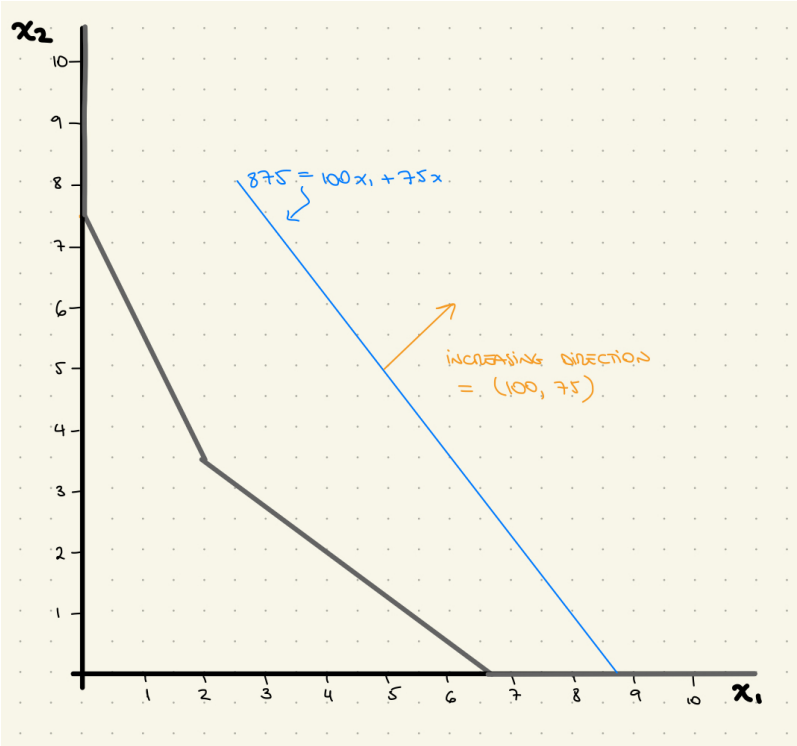
We may have thought that, since we removed some constraints, the solution would change. However, observe that we did not remove any of the constraints that intersect at the optimal solution. Intuitively, the constraints we removed were not really limiting the optimal solution.

(b) Now the objective function changed and, hence, the increasing/decreasing direction also changed. Solving graphically, we obtain the following solution:



In this case, all the points in the line highlighted in green are optimal and have the same objective function value. We observe these situations when the decreasing direction is perpendicular to one of the constraints in a minimization problem. Similarly, in a maximization problem we observe this situation when the increasing direction is perpendicular to one of the constraints.

(c) In the last case we have the same expression for the objective function as in part (a), but we are maximizing instead of minimizing. Then, we are interested in the increasing direction instead of the decreasing direction. We solve graphically and we obtain the following:



Observe that, no matter how much we move our objective function curve to the right, there will always be a better solution. In this case, we say that the problem is unbounded and the optimal value is ∞. In minimization problems, unbounded problems have optimal value $-\infty$.

□

In the example above we showed different situations that we may face when solving optimization problems. Observe that infeasibility is a consequence of the constraints being contradictory. The number of optimal solutions (0,1, or multiple), on the other hand, depends on the objective function. In the example above, we showed that the same feasible region may lead to a unique, multiple or no optimal solution depending on the properties of the objective function and how it relates to the feasible region.

2.3 Large-scale optimization models and indexing

So far, we have constructed mathematical models for very small problems. Both, the Two Crude Petroleum example and the example of fencing a rectangular area only had 2 variables. However, real-life problems are much larger. In this section we will learn a convenient way to write these models. Our main goal is to write mathematical models in a concise way, so that analyzing them is easier. Similarly to the previous section, we work on an example as we learn the new concepts.

Example 2.5 (Application 2.2 from the textbook)**.** *Pi Hybrids is a large manufacturer of corn seeds. They operate ℓ facilities producing seeds of m hybrid corn varieties and distribute them to customers in n sales regions. They want to know how to carry out these production and distribution operations at minimum cost.*

After some effort, a variety of parameters that we will take as constant have been estimated:

- *The cost per bag of producing each hybrid at each facility*
- *The cost per bag of shipping each hybrid from each facility to each customer region*
- *The corn processing capacity of each facility in bushels*
- *The number of bushels of corn that must be processed to make a bag of each hybrid*
- *The number of bags of each hybrid demanded in each customer region*

Our task is to produce a suitable optimization model.

To write problems concisely, we group similar variables under the same symbol, and we add an **index** to imply that there are many similar quantities. We may use more than one index to characterize all the **dimensions** of the variables that we are interested in. The dimensions represent how we classify our variables and parameters, and we frequently have more than 1. For example, in the Pi Hybrids problem, we classify according to the facility, the hybrid variety and the sales region. Then, we use the following sets of indices:

$$\begin{aligned} f &= \text{ production facility number} \\ h &= \text{ hybrid variety number} \\ r &= \text{ sales region number} \end{aligned}$$

With these indices, we may talk about the production of hybrid h at facility f, to be sold at region r. Observe that f takes values in the set $\{1, \ldots, \ell\}$, h in the set $\{1, \ldots, m\}$ and r in $\{1, \ldots, n\}$.

Now that we decided on the indices we will use, we are ready to define our decision variables. Pi Hybrid needs to decide the production of each hybrid at each facility, and the number of bags of each hybrid that each facility will ship to each sales area. Hence, we define the following decision variables:

$$\begin{aligned} x_{f,h} \triangleq\ & \text{Number of bags of hybrid } h \text{ produced at facility } f \\ & f \in \{1, \ldots, \ell\},\ h \in \{1, \ldots, m\}. \end{aligned}$$

$$\begin{aligned} y_{f,h,r} \triangleq\ & \text{Number of bags of hybrid } h \text{ shipped from facility } f \text{ to sales region } r \\ & f \in \{1, \ldots, \ell\},\ h \in \{1, \ldots, m\},\ r \in \{1, \ldots, n\}. \end{aligned}$$

Observe that we defined a very large number of decision variables using only 2 symbols. There are ℓm variables x and ℓmn variables y, which gives a total of $\ell m + \ell mn$ variables. If $\ell = 20$, $m = 25$ and $n = 30$, we get 500 x-variables and 15000 y-variables. Hence, we have 15500 variables in total.

Recall the list of parameters that Pi Hybrid estimated, and observe they are a lot too! Hence, we introduce symbols to denote them, and we associate these symbols with the corresponding indices. Define the following symbolic names for each of the parameters. For each $f \in \{1, \ldots, \ell\}$, $h \in \{1, \ldots, m\}$ and $r \in \{1, \ldots, n\}$, let

$$\begin{aligned} p_{f,h} &\triangleq \text{ Cost per bag of producing hybrid } h \text{ at facility } f \\ u_f &\triangleq \text{ Corn processing capacity of facility } f \text{ in bushels} \\ a_h &\triangleq \text{ Number of bushels of corn that must be processed to obtain a bag of hybrid } h \\ d_{h,r} &\triangleq \text{ Number of bags of hybrid } h \text{ demanded at region } r \\ s_{f,h,r} &\triangleq \text{ Cost per bag of shipping hybrid } h \text{ from facility } f \text{ to sales region } r \end{aligned}$$

Now we can write the objective function using our indexed notation and summations over the indices. The goal is to minimize the total cost, and we know that the total cost is the total production cost plus the total shipping cost. Then, the objective function is

$$\min \quad \sum_{f=1}^{\ell} \sum_{h=1}^{m} p_{f,h} x_{f,h} + \sum_{f=1}^{\ell} \sum_{h=1}^{m} \sum_{r=1}^{n} s_{f,h,r} y_{f,h,r}.$$

Observe that, since we are computing the total cost, we need to consider all the hybrids, all the facilities and all the sales regions. Then, we sum over all the indices that are relevant for each decision variable. Here, it is key to realize that the word **total** is associated to a **sum over indices**.

In the case of the constraints, we can detect groups of similar constraints. For example, each facility needs to produce a number of bushels below its capacity. Since there are ℓ facilities, there are also ℓ constraints. However, they all represent capacity of a facility, so we may group them and use the index to indicate that all the facilities must satisfy a similar constraint. More explicitly, we write

$$\text{Total production of facility } f \leq \text{Capacity of facility } f, \quad \forall f \in \{1, \ldots, \ell\}, \tag{1}$$

where the symbol $\forall$ reads "for all". Equation (1) is written in a single line, but it represents ℓ constraints. Here, we associate the word **"each"** with the **"for all"** symbol.

Now let's write an actual mathematical expression for Equation (1). The total production of facility f is the sum of the production of all the hybrids, and the production of hybrid h at facility f corresponds to the parameter a_h times $x_{f,h}$. Then, (1) can be written as

$$\sum_{h=1}^{m} a_h x_{f,h} \leq u_f \qquad \forall f \in \{1, \ldots, \ell\},$$

where we computed the *total* over hybrids, for *each* facility.

Similarly to the capacity constraint, we can write the demand satisfaction constraint as follows. Each sales region has a specific demand for each hybrid. Then, the total amount of each hybrid that is shipped to each region must be at least equal to the demand, where the total amount of each hybrid is the sum of the production in all the facilities. Then, the demand satisfaction family of constraints can be written as

$$\sum_{f=1}^{\ell} y_{f,h,r} \geq d_{h,r} \qquad \forall h \in \{1, \ldots, m\},\ r \in \{1, \ldots, n\}$$

We are almost ready with the main constraints of the Pi Hybrids problem. Observe that the variables x and y are related by definition. However, we have to "tell the optimization software" how they are related. It is not sufficient to describe the variables in words because the software does not understand English. We need to explicitly include a constraint that models this relationship. Recall that $x_{f,h}$ is the production of hybrid h at facility f and $y_{f,h,r}$ is the production of hybrid h from facility f that is shipped to region r. Then, we must have

$$\sum_{r=1}^{n} y_{f,h,r} = x_{f,h} \qquad \forall f \in \{1, \ldots, \ell\},\ h \in \{1, \ldots, m\}.$$

We call this constraint a **balance constraint**.

Therefore, the Pi Hybrid problem can be represented by the following optimization model:

$$\begin{aligned}
\min \quad & \sum_{f=1}^{\ell} \sum_{h=1}^{m} p_{f,h} x_{f,h} + \sum_{f=1}^{\ell} \sum_{h=1}^{m} \sum_{r=1}^{n} s_{f,h,r} y_{f,h,r} \\
\text{s.t.} \quad & \sum_{h=1}^{m} a_h x_{f,h} \leq u_f \qquad \forall f \in \{1, \ldots, \ell\} \qquad \text{(production capacity)} \\
& \sum_{f=1}^{\ell} y_{f,h,r} = d_{f,h} \qquad \forall h \in \{1, \ldots, m\},\ r \in \{1, \ldots, n\} \qquad \text{(demand satisfaction)} \\
& \sum_{r=1}^{n} y_{f,h,r} = x_{f,h} \qquad \forall f \in \{1, \ldots, \ell\},\ h \in \{1, \ldots, m\} \qquad \text{(balance)} \\
& x_{f,h} \geq 0 \qquad \forall f \in \{1, \ldots, \ell\},\ h \in \{1, \ldots, m\} \qquad \text{(nonnegativity)} \\
& y_{f,h,r} \geq 0 \qquad \forall f \in \{1, \ldots, \ell\},\ h \in \{1, \ldots, m\},\ r \in \{1, \ldots, n\} \qquad \text{(nonnegativity)}
\end{aligned}$$

Observe that we wrote the optimization model in only a few lines, but there are ℓ production capacity constraints, mn demand satisfaction constraints, ℓm balance constraints and $\ell m + \ell mn$ nonnegativity constraints. With the numbers we gave before ($\ell = 20$, $m = 25$, $n = 30$) this gives a total of 20 capacity, 750 demand, 500 balance and 15500 nonnegativity constraints, i.e., we wrote 16,770 constraints in 5 lines.

2.4 Linear and nonlinear programs

In Section 2.2 we learned how to solve problems with two or three decision variables. However, real-life problems are much larger and, hence, more complicated to solve. Among all the types of optimization problems, linear programs are the easiest to solve. We will learn why later, and now we will focus on detecting linear and nonlinear programs.

Before defining what a linear problem is, let's introduce some notation for any optimization problem.

Principle 2.2 (Principle 2.27 from the textbook)**.** *The general form of a mathematical program or single-objective optimization model is:*

$$\begin{aligned} \min \text{ or } \max \quad & f(x_1, \ldots, x_n) \\ s.t. \quad & g_i(x_1, \ldots, x_n) \big[\leq, \geq, \text{ or } = \big] b_i \qquad \forall i \in \{1, \ldots, m\}, \end{aligned}$$

where f, $g_1, \ldots, g_m$ are given functions of the decision variables $x_1, \ldots, x_n$, and $b_1, \ldots, b_m$ are specified constant parameters.

We call the general form from the principle above, the general functional form of a problem. Observe that everything related to the decision variables is absorbed by the functions f and g_i, and we only leave constant parameters on the right hand side of the constraints.

Example 2.6. *Write the following optimization problem in the general functional form, assuming that w_1, w_2, w_3 are the decision variables.*

$$\begin{aligned} \max \quad & w_1^2 + 8w_2 + w_3^2 \\ s.t. \quad & w_1 + 6w_2 \leq 10 + w_2 \\ & w_2^2 = 7 \\ & w_1 \geq w_3 + a \\ & w_1, w_2 \geq 0 \end{aligned}$$

Solution. Here the variables are represented by $\boldsymbol{w}$, that is, the letter a in the third constraint represents a parameter. Therefore, we obtain the following functions

$$\begin{aligned} f(w_1, w_2, w_3) &= w_1^2 + 8w_2 + w_3^2 \\ g_1(w_1, w_2, w_3) &= w_1 + 5w_2 \\ g_2(w_1, w_2, w_3) &= w_2^2 \\ g_3(w_1, w_2, w_3) &= w_1 - w_3 \\ g_4(w_1, w_2, w_3) &= w_1 \\ g_5(w_1, w_2, w_3) &= w_2 \end{aligned}$$

and the right-hand side parameters

$$b_1 = 10,\ b_2 = 7,\ b_3 = a,\ b_4 = 0,\ b_5 = 0.$$

□

Now that we have set up the notation, we can define linear functions and problems.

Definition 2.5 (Def 2.28 from the textbook)**.** *A function is linear if it is constant-weighted sum of the decision variables. Otherwise, it is nonlinear.*

Example 2.7. *Assuming that x_1, x_2, x_3 are the only decision variables, determine if the following functions are linear:*

(a) $f(x_1, x_2, x_3) = 9x_1 - 17x_3$

(b) $f(x_1, x_2, x_3) = \dfrac{c_1}{x_1} + c_2 x_2 - 6x_3$

(c) $f(x_1, x_2, x_3) = e^{\alpha} x_1 + \ln(\beta) x_2$

(d) $f(x_1, x_2, x_3) = \alpha e^{x_1} + \beta \ln(x_2)$

(e) $f(x_1, x_2, x_3) = \dfrac{x_1 + x_2}{x_1 - x_3}$

Solution.

(a) Linear

(b) Nonlinear

(c) Linear

(d) Nonlinear

(e) Nonlinear. However, observe that if this function is part of a constraint, we can easily linearize it. For example, if we had

$$\frac{x_1 + x_2}{x_1 - x_3} \leq 3$$

we can rewrite it in the following linear form:

$$x_1 + x_2 - 3x_1 + 3x_3 \leq 0 \qquad \Longleftrightarrow \qquad -2x_1 + x_2 + 3x_3 \leq 0.$$

□

We are not only interested in linear functions. In this course we will study linear problems, which are composed of linear functions and a few more characteristics.

Definition 2.6. *An optimization problem with decision variables $\boldsymbol{x} = (x_1, \ldots, x_n)$ and m constraints is <u>linear</u> if all the functions $f(\boldsymbol{x})$ and $g_i(\boldsymbol{x})$ for all $i \in \{1, \ldots, m\}$ are linear functions of $\boldsymbol{x}$ and the decision variables are continuous.*

Observe that when we introduced the notation $f(\boldsymbol{x})$ and $g_i(\boldsymbol{x})$ for the objective function and the constraints, we assummed that the constraints are closed inequalities $(\leq, \geq, =)$. This assumption is also valid when we define linear problems, that is, a linear problem must have only closed inequalities. Let's see some examples.

Example 2.8. *Determine whether the following optimization problems are linear. In case they are not, show why.*

(a) If the decision variables are x and y, consider

$$\begin{aligned} \min \quad & x + 3y \\ s.t. \quad & x + y \geq 10 \\ & 2x + \frac{1}{2}y = 5 \\ & x, y \geq 0 \end{aligned}$$

(b) If the decision variables are x, y, z, consider

$$\begin{aligned} \min \quad & x + 2y + z \\ s.t. \quad & 2x + y - z \leq 20 \\ & x, y, z \geq 0 \\ & x \in \mathbb{Z} \end{aligned}$$

(c) If the decision variables are x, y, z, consider

$$\begin{aligned} \max \quad & \alpha_1 x + \log(\alpha_2) y - z \\ s.t. \quad & x + y < z \\ & x, y, z \geq 0 \end{aligned}$$

(d) If the decision variables are x, y, z, consider

$$\begin{aligned} \min \quad & |x - 5| + |y - 3| + |z - 10| \\ s.t. \quad & 2x + y + z \leq 20 \end{aligned}$$

Solution.

(a) It is a linear problem.

(b) It is not a linear problem because the variable x is not continuous.

(c) Even though all the functions $f(x, y, z)$ and $g_i(x, y, z)$ are linear, this problem is not linear because the first constraint is strict.

(d) No because the objective function is nonlinear, as it has absolute values. However, we will see later that we will be able to linearize this problem, that is, rewrite in an equivalent way where all the functions are linear. The key factor is that $|x| = \max\{x, -x\}$ and we are minimizing. Therefore, the objective function is of the form min max.

□

3 Improving Search

[This section is based on Chapter 3 of the textbook]

In this chapter we will learn the foundation of many optimization algorithms. We already learned the graphical method, which helped us develop some intuitions. However, this method is only useful if we can graph the constraints and contour curves of the objective function. Hence, we can only use it in problems with two or three variables. In practice, most of the problems have hundreds or thousands of variables, so we need another approach.

In your calculus classes, you probably learned how to compute the global maximum and minimum of a function and how the approach changes if we have a bounded interval where each of the variables live. All of these have a close-form solution, that is, we can solve it analytically.

In this chapter, instead, we will learn the foundations of solving optimization problems numerically. In simple words, we will iteratively search for an optimal solution. That is, we will start at some feasible solution, compare it to others, and move to a better solution while satisfying the constraints, until we find one that is optimal. This is called improving search, or local search.

3.1 Improving search, local, and global optima

Let's start with some definitions and conventions on notation.

Definition 3.1 (Definition 3.1 from the textbook)**.** *A solution is a choice of values for all the decision variables.*

For example, in the Two Crude example, a solution would be $x_1 = 3$, $x_2 = 2$. Observe that a solution may satisfy the constraints (in which case it is a feasible solution), or not.

As learned in Section 2, many of the problems we model have multiple variables. Hence, we represent a solution by a vector. If our problem has n variables, then every solution will be an n-dimensional vector. In this note, we will use bold letters to represent vectors, but when we write by hand we will add an arrow on top of the variable. For example, the solution for the Two Crude example above can be represented as $\boldsymbol{x} = (3, 2)$ and if we write this example on the whiteboard (or your homework/workshop/exam) you may write $\vec{x} = (3, 2)$.

As mentioned in the introduction of this chapter, we will iteratively solve our problems. Hence, we need notation to distinguish the sequence of solutions that we visit.

Definition 3.2 (Definition 3.2 from the textbook)**.** *For a model with decision vector $\boldsymbol{x}$, the first solution visited by a search is denoted by $\boldsymbol{x}^{(0)}$, the next $\boldsymbol{x}^{(1)}$, an so on. In general, we write $\boldsymbol{x}^{(t)}$ to denote the $(t+1)^{th}$ solution we visit.*

Let's see an example of how we use this notation.

Example 3.1 (Example 3.1 from the textbook)**.** *The following table shows the sequence of solutions encountered by an improving search of an optimization model with four variables.*

Iteration	x_1	x_2	x_3	x_4
Iteration 1	*1*	*0*	*1*	*2*
Iteration 2	*1*	*1*	*-2*	*4*
Iteration 3	*2*	*1*	*-1*	*4*
Iteration 4	*5*	*1*	*-1*	*6*

(a) Express the solutions in the notation introduced above

(b) Identify the value of $x_1^{(3)}$ and $x_3^{(1)}$

Solution.

(a) We have:

$$\boldsymbol{x}^{(0)} = (1, 0, 1, 2)$$
$$\boldsymbol{x}^{(1)} = (1, 1, -2, 4)$$
$$\boldsymbol{x}^{(2)} = (2, 1, -1, 4)$$
$$\boldsymbol{x}^{(3)} = (5, 1, -1, 6)$$

(b) Observe that $x_1^{(3)}$ is the first element of the vector corresponding to the fourth iteration, that is, $x_1^{(3)} = 5$. On the other hand, $x_3^{(1)}$ is the third element of the vector corresponding to the second iteration, that is, $x_3^{(1)} = -2$. iteration

□

Now that we have established some notation, we can start learning what is an improving search algorithm, and how we use them.

Definition 3.3 (Definition 3.3 from the textbook)**.** *Improving searches are numerical algorithms that begin at a feasible solution to a given optimization model and advance along a search path of feasible points with ever-improving objective function value.*

In other words, when we run an improving search algorithm, we search through the feasible set and we make sure that each iteration improves the objective function value. In minimization/maximization models, this improvement means that the value of the objective function in consecutive iterations will decrease/increase.

The intuition of an improving search is simple. An example of an improving search is the graphical method introduced in the previous chapter. However, when our optimization models have more than three variables we cannot use a graph and, hence, our "visibility" for better solutions is limited. Hence, we search in nearby points.

In the following definition we formalize this concept, and we introduce two notions of optimal solutions.

Definition 3.4 (Definitions 3.4, 3.5 and 3.7 from the textbook)**.**

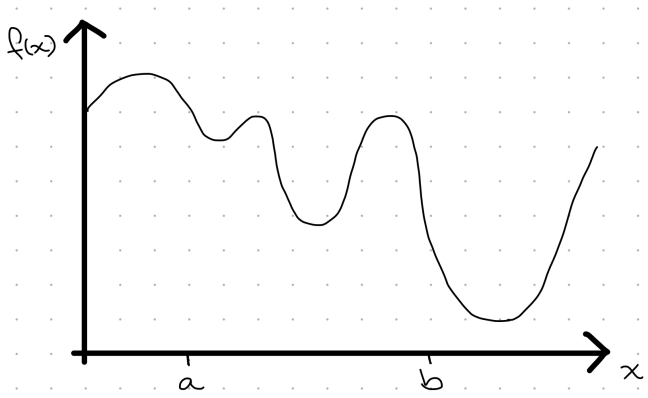
- *The neighborhood of a current solution $\boldsymbol{x}^{(t)}$ consists of all nearby points, that is, all points within a small distance of $\boldsymbol{x}^{(t)}$*
- *A solution is a local optimum if it is feasible and if sufficiently small neighborhoods surrounding it contain no points that are both feasible and have better objective function value.*
- *A solution is a global optimum if it is feasible and no other feasible solution has superior objective value.*

Let's see some examples.

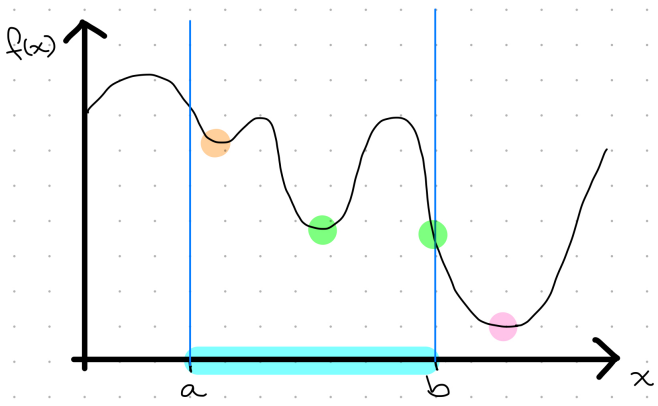
Example 3.2. *Consider an optimization problem with one variable, of the form*

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & a \leq x \leq b \end{aligned}$$

In the following picture, we show a graph of $f(x)$. Show the feasible region in the figure, and mark all the local and global optima.



Solution. Let's use the following marked areas from the picture to answer the question:



We have:

- The light blue area is the feasible region. We additionally draw vertical lines that intersect $f(x)$ at $x = a$ and $x = b$ to facilitate analyzing which optimal solutions are feasible and which are not.
- The orange circle shows a local minimum. Indeed, in a very small neighborhood of this point, all the values of $f(x)$ are higher. However, if we look at the entire graph, this local optimum is not a global optimum.
- The green circles show two points that have the same objective function value. Further, they are local and global optima because **in the feasible region** they have the smallest objective function values.
- The pink circle shows a point where $f(x)$'s value is smaller than all the values pointed above. However, this pink circle is **outside** the feasible region, so it is **not** a local or global optimum.

□

The example above intuitively shows the following principle.

Principle 3.1 (Principles 3.8 and 3.9 from the textbook)**.** *Global optima are always local optima, but local optima may not be global optima.*

Indeed, the definition of global optimum compares a solution against the entire feasible region, whereas local optima are only compared against feasible neighbors.

In an ideal world we always want to compute a global optimum. However, we cannot make sure we reached a global optimum (instead of a local optimum) unless we know the structure of the whole feasible region and the objective function. The example above shows an easy feasible region, but complicated objective function. Unfortunately, many problems have complicated objective and complicated feasible region. Hence, computing global optima becomes intractable.

But no worries! There is still a lot of hope! In many optimization problems (such as linear problems), a local optimum is always a global optimum. Therefore, we only need a method to find a local optimum. In problems where local optima are not necessarily global we need to do some extra work. For example, we may solve the problem multiple times, making sure that each time we start from a different feasible solution. Then, we may find different local optima and we can compare them. Unless we explore the entire feasible region, we won't be sure if we reached a global optimum, but at least we know we have a "good local optimum." Such local optimum is frequently called heuristic optimum or approximate optimum.

3.2 Search with improving and feasible directions

Now we know the main ideas, so let's learn how to do it! We've been talking about iterations, but how do we choose the next point? In general, moving from one point to another requires two choices: a direction to move and a step size. The direction will tell us where we are looking to move, and the step size how much we go in that direction. Let's start with some key concepts and ideas.

Definition 3.5 (Definition 3.12 from the textbook)**.** *Improving searches advance from the current solution* $\boldsymbol{x}^{(t)}$ *to the new solution* $\boldsymbol{x}^{(t+1)}$ *as*

$$\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} + \lambda_{t+1}\Delta\boldsymbol{x}^{(t+1)}$$

The vector $\Delta \boldsymbol{x}^{(t+1)}$ defines a move direction at $\boldsymbol{x}^{(t)}$, and the scalar multiplier $\lambda_{t+1} > 0$ defines the step size, which indicates how far we pursue the direction.

Let's see an example.

Example 3.3 (based on Examples 3.3 and 3.4 from the textbook)**.**

(a) An improving search beginning at $\boldsymbol{w}^{(0)} = (5, 1)$ employs first move direction $\Delta \boldsymbol{w}^{(1)} = (0, 1)$ for step $\lambda_1 = 1/3$, and then $\Delta \boldsymbol{w}^{(2)} = (2, 0)$ with step $\lambda_2 = 4$. Determine the solutions visited and show the path followed in a graph.

(b) The first solutions visited by an improving search are $\boldsymbol{y}^{(0)} = (5, 11)$, $\boldsymbol{y}^{(1)} = (4, 9)$ and $\boldsymbol{y}^{(2)} = (0, 8)$. Graph the path and determine the moving directions employed assuming unit step size.

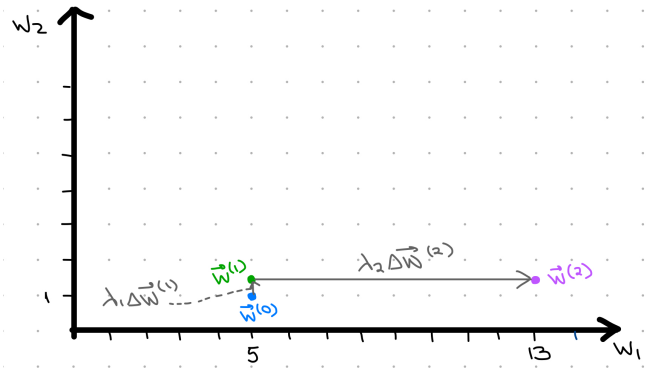
Solution.

(a) We first compute the solutions $\boldsymbol{w}^{(1)}$ and $\boldsymbol{w}^{(2)}$. We obtain:

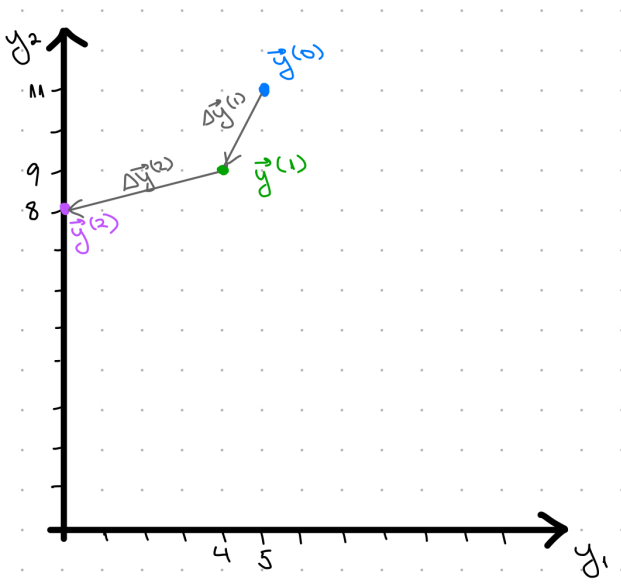
$$\boldsymbol{w}^{(1)} = \boldsymbol{w}^{(0)} + \lambda_1 \Delta \boldsymbol{w}^{(1)} = (5, 1) + \frac{1}{3}(0, 1) = \left(5, \frac{4}{3}\right)$$

$$\boldsymbol{w}^{(2)} = \boldsymbol{w}^{(1)} + \lambda_2 \Delta \boldsymbol{w}^{(2)} = \left(5, \frac{4}{3}\right) + 4(2, 0) = \left(13, \frac{4}{3}\right)$$

We obtain the following graph:



(b) We obtain the following graph:



and the move directions assuming a step size $\lambda = 1$ are:

$$\Delta \boldsymbol{y}^{(1)} = \boldsymbol{y}^{(1)} - \boldsymbol{y}^{(0)} = (4, 9) - (5, 11) = (-1, -2)$$
$$\Delta \boldsymbol{y}^{(2)} = \boldsymbol{y}^{(2)} - \boldsymbol{y}^{(1)} = (0, 8) - (4, 9) = (-4, -1)$$

□

Now that we know how to move in a direction to the next iteration, we need to learn how we choose a direction. Recall that the whole goal of our iterations is finding an optimal solution, and we want to improve the solution in every step. Hence, the direction we choose should improve the objective function value. Additionally, we must reach a feasible solution after taking the current step. Below we define these two features of a moving direction.

Definition 3.6 (Definitionis 3.13 and 3.14 from the textbook)**.** *The vector $\Delta \boldsymbol{x}$ is:*

(i) an improving direction at the current solution $\boldsymbol{x}^{(t)}$ if the objective function value at $\boldsymbol{x}^{(t)} + \lambda \Delta \boldsymbol{x}$ is better than the objective function value at $\boldsymbol{x}^{(t)}$ for all $\lambda > 0$ sufficiently small.

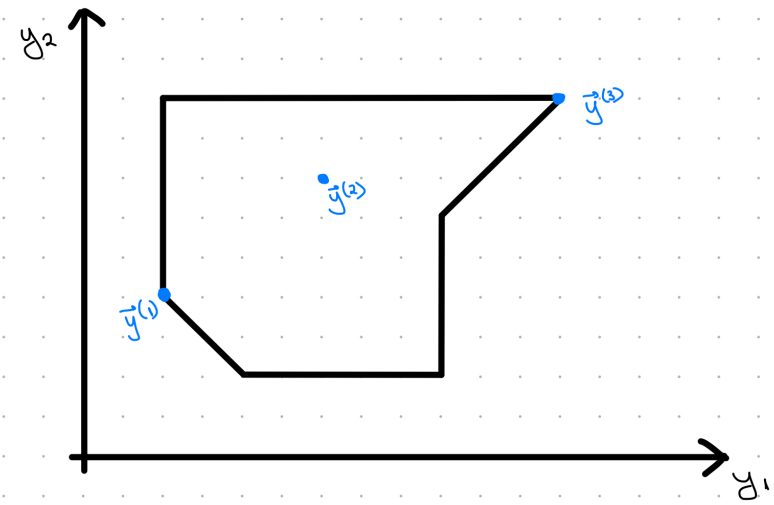
(ii) a feasible direction at the current solution $\boldsymbol{x}^{(t)}$ if the point $\boldsymbol{x}^{(t)} + \lambda \Delta \boldsymbol{x}$ violates no model constraint (is feasible) for $\lambda > 0$ sufficiently small.

Observe that both definitions consider small λ because we are searching for local optima. Let's see an example.

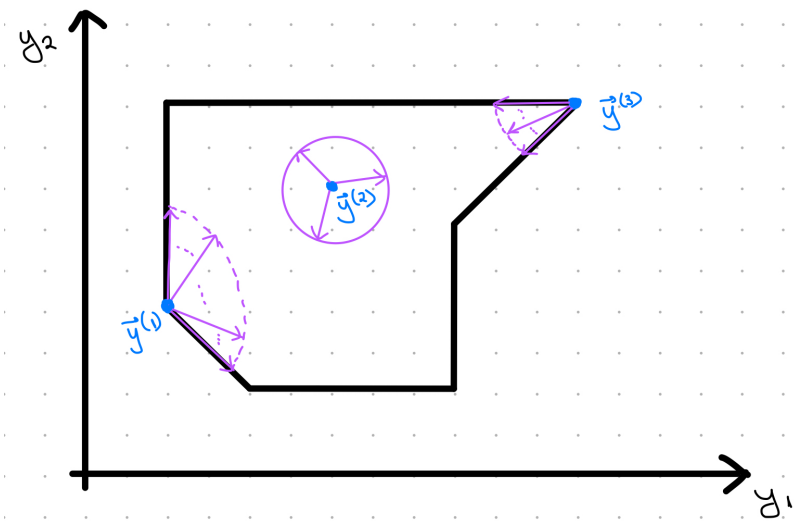
Example 3.4. *Consider the following optimization problem*

$$\begin{aligned} \min \quad & y_2 \\ s.t. \quad & \boldsymbol{y} \in \mathcal{P} \end{aligned}$$

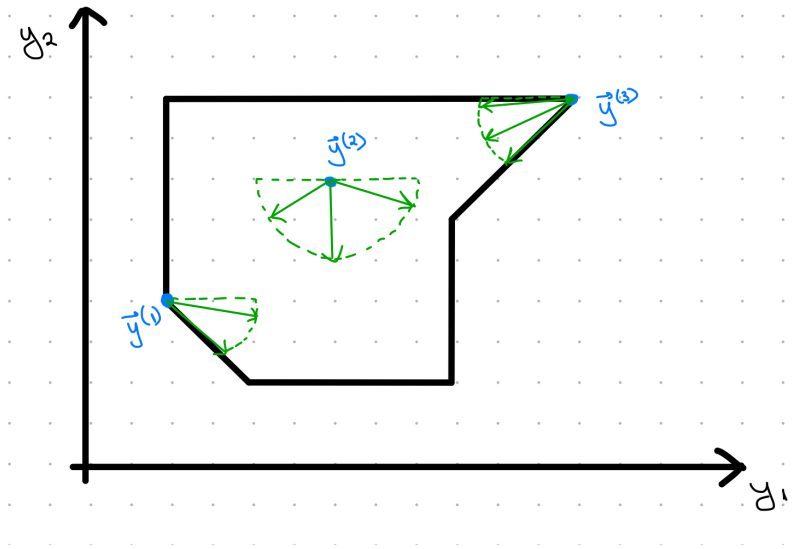
where $\mathcal{P}$ is the polygon from the figure. Determine all the feasible and the improving directions starting from the points $\boldsymbol{y}^{(1)}$, $\boldsymbol{y}^{(2)}$ and $\boldsymbol{y}^{(3)}$.



Solution. We first show all the feasible directions in the graph. The dashed lines show the feasible directions range, and we additionally show a few examples (arrows).



Since the objective function is linear, the improving directions are the same in all the points, that is, all the directions for which the value of y_2 decreases. In the following graph we show the feasible and improving directions:



□

We just learned what is a "good direction" in an improving search. Our next question is about the step size, that is, how much can we go in a feasible improving direction.

Principle 3.2 (Principle 3.15 from the textbook)**.** *Improving searches normally apply the maximum step $\lambda > 0$ for which the selected move direction continues to retain feasibility and improve the objective function.*

Intuitively, if we find a good direction (feasible and improving), we should make the most out of it. Hence, the step size will be as large as possible under the condition that the direction is still "good."

Termination condition

Observe that our definition of "good" move directions will most likely take us to a local optimum. But, how do we know that we found it? In the following two principles we answer this question.

Principle 3.3 (Principle 3.16 from the textbook)**.** *No optimization model solution at which an improving feasible direction is available can be a local optimum.*

In other words, if there is at least one improving direction, we know that the objective function value can be improved in a neighborhood. Hence, we have not reached a local optimum.

Principle 3.4 (Principle 3.17 from the textbook)**.** *Under mild regularity conditions, when a continuous improving search terminates at a solution admitting no improving feasible direction, the point is a local optimum.*

In other words, if we cannot compute an improving direction, we must be at a local optimum. In this course we won't dwell into the regularity conditions stated above. In most cases, these conditions are satisfied, so we will take them for granted.

So far, we are assuming that we are able to compute a local optimum. However, this may not be the case. For example, consider the following optimization problem

$$\begin{aligned} \max \quad & x_1 + x_2 \\ \text{s.t.} \quad & x_1 \geq 0 \end{aligned}$$

In this case, there is no local or global optimum because the objective function can grow to ∞ without breaking any constraint. Such models are called unbounded.

Definition 3.7 (Definition 2.20 from the textbook)**.** *An optimization model is unbounded when feasible choices of the decision variables can produce arbitrarily good objective function values.*

In terms of an improving search algorithm, unboundedness means that an improving feasible direction remains improving and feasible no matter how large the step size λ is. That is, if we conclude $\lambda = \infty$, then the problem is unbounded.

In the example above, suppose $\boldsymbol{x}^{(0)} = (3, 1)$. Then, one of the (many) improving feasible directions is $\Delta\boldsymbol{x} = (1, 1)$. For step size λ, the next point $\boldsymbol{x}^{(1)}$ is:

$$\boldsymbol{x}^{(1)} = (3 + \lambda, 1 + \lambda)$$

which will not break any constraint for $\lambda > 0$.

Additionally, it is easy to see that $\Delta\boldsymbol{x}$ is always an improving direction. In fact, the change in the objective function value from $\boldsymbol{x}^{(0)}$ to $\boldsymbol{x}^{(1)}$ is:

$$\left(x_1^{(1)} - x_2^{(1)}\right) - \left(x_1^{(0)} + x_2^{(0)}\right) = (3 + \lambda + 1 + \lambda) - (3 + 1) = 2\lambda$$

The value of 2λ is always positive, which means that the objective function value of $\boldsymbol{x}^{(1)}$ is always larger than the objective value of $\boldsymbol{x}^{(0)}$. Since the problem is a maximization, it means that for all $\lambda > 0$, $\boldsymbol{x}^{(1)}$ is better than $\boldsymbol{x}^{(0)}$. Hence, $\Delta\boldsymbol{x} = (1, 1)$ is always an improving direction.

Since we can choose λ as large as we want (even ∞) without breaking feasibility, our improving search detects unboundedness.

As a summary of this section, we developed continuous improving search algorithms as shown below.

Algorithm 3.1 (Continuous Improving Search)**.**

0. ***Initialization:*** *Choose any starting feasible solution $\boldsymbol{x}^{(0)}$ and set the solution index $t \leftarrow 0$.*
1. ***Local optimality:***
   - *If no improving feasible direction $\Delta\boldsymbol{x}$ exists at the current solution $\boldsymbol{x}^{(t)}$, STOP. Current solution $\boldsymbol{x}^{(t)}$ is a local optimum (under mild regularity conditions).*
   - *Otherwise, continue to step 2.*
2. ***Move direction:*** *Construct an improving feasible direction at current solution $\boldsymbol{x}^{(t)}$ and call it $\Delta\boldsymbol{x}^{(t+1)}$*
3. ***Step size:*** *Compute maximum $\lambda_{t+1} > 0$ such that the direction $\Delta\boldsymbol{x}^{(t+1)}$ continues to both improve and retain feasibility.*
   - *If such λ_{t+1} does not exist (that is, can be $+\infty$), STOP. Model is unbounded.*
   - *Otherwise, continue to step 4.*

4. ***Update current solution:*** *Compute the next solution as*

$$\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} + \lambda_{t+1}\Delta\boldsymbol{x}^{(t+1)},$$

update index $t \leftarrow t+1$ *and go back to step 1.*

This is a generic algorithm, but it has all the pieces an improving search needs. In this course we will see multiple improving search algorithms, and you will notice that all of them have the same structure. The only difference is how to choose the improving directions and step sizes. In the next subsection we show and example of how to make those choices.

3.3 Algebraic conditions for improving and feasible directions

So far in this chapter we have focused on understanding improving searches geometrically. In this section we learn how to compute the iterations. We start with the computation of feasible improving directions.

Among all the improving directions at a given point, we choose the direction of most rapid improvement. Recall from your multi-variate calculus class, that the gradient of a multi-variate function $f(x_1, x_2, \ldots, x_n)$ is the vector of partial derivatives, that is,

$$\nabla f(\boldsymbol{x}) = \left(\frac{\partial}{\partial x_1} f(\boldsymbol{x}),\ \frac{\partial}{\partial x_2} f(\boldsymbol{x}), \cdots, \frac{\partial}{\partial x_n} f(\boldsymbol{x})\right)$$

Further, the gradient at a given point is a linear approximation of the function in a small neighborhood. Further, we have the following principle.

Principle 3.5 (Principle 3.20 from the textbook)**.** *Gradients show graphically as vectors perpendicular to contours of the objective function and point in the direction of most rapid value increase.*

The principle above suggests that we should choose the gradient as the move direction. However, the gradient may not be a feasible direction. In the following principle we use the gradient to show that other move directions are improving.

Principle 3.6 (based on Principles 3.21 and 3.22)**.** *The direction* $\Delta\boldsymbol{x}$ *is improving for a maximize/minimize objective function* $f(\cdot)$ *at a point* $\overline{\boldsymbol{x}}$ *if* $\nabla f(\overline{\boldsymbol{x}})^T\Delta\boldsymbol{x} > 0$/$\nabla f(\overline{\boldsymbol{x}})^T\Delta\boldsymbol{x} < 0$*. If* $\nabla f(\overline{\boldsymbol{x}})^T\Delta\boldsymbol{x} = 0$*, we cannot solve without further information.*

Let's see a sketch proof for the maximization case. The minimization case is analogous.

Proof. Suppose we start at the point $\overline{\boldsymbol{x}}$ and we want to move to the point $\overline{\boldsymbol{x}}' = \overline{\boldsymbol{x}} + \lambda\Delta\boldsymbol{x}$, where $\lambda > 0$. Then, the change in the objective function value is:

$$\begin{aligned}
f(\overline{\boldsymbol{x}}') - f(\overline{\boldsymbol{x}}) &= \frac{1}{n}\sum_{j=1}^{n}\left(\frac{f(\overline{\boldsymbol{x}}') - f(\overline{\boldsymbol{x}})}{\overline{x}'_j - \overline{x}_j}\right)(\overline{x}'_j - \overline{x}_j) \\
&= \frac{1}{n}\sum_{j=1}^{n}\left(\frac{f(\overline{\boldsymbol{x}} + \lambda\Delta\boldsymbol{x}) - f(\overline{\boldsymbol{x}})}{\lambda\Delta x_j}\right)\lambda\Delta x_j \qquad \text{(def. of } \overline{\boldsymbol{x}}'\text{)} \\
&\approx \frac{1}{n}\sum_{j=1}^{n}\left(\frac{\partial f}{\partial x_j}\right)\lambda\Delta x_j \qquad \text{(for small } \Delta x_j\text{)} \\
&= \frac{\lambda}{n}\nabla f(\overline{\boldsymbol{x}})^T\Delta\boldsymbol{x}
\end{aligned}$$

Hence, the sign of $\nabla f(\overline{\boldsymbol{x}})^T\Delta\boldsymbol{x}$ determines if a direction is improving or not. □

Let's see an example.

Example 3.5 (Example 3.9 from the textbook)**.** *Determine if the following directions are improving or show why further information is required.*

(a) $\Delta\boldsymbol{w} = (1, 0, -2)$ *for minimize* $f(\boldsymbol{w}) = w_1^2 + 5w_2w_3$ *at* $\overline{\boldsymbol{w}} = (2, 1, 0)$.

(b) $\Delta \boldsymbol{y} = (3, -6)$ *for maximize* $f(\boldsymbol{y}) = 9y_1 + 40y_2$ *at* $\overline{\boldsymbol{y}} = (13, 2)$.

(c) $\Delta \boldsymbol{z} = (-6, 2)$ *for minimize* $f(\boldsymbol{z}) = 5z_1^2 - 3z_1 z_2 + z_2^2$ *at* $\overline{\boldsymbol{z}} = (1, 3)$.

Solution. In each case, we compute the gradient, we evaluate at the current point and then take the dot product of the gradient and the direction.

(a) The gradient is:

$$\begin{aligned} \nabla f(\boldsymbol{w}) &= \left(\frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}, \frac{\partial f}{\partial w_3} \right) = (2w_1, 5w_3, 5w_2) \\ \implies \nabla f(\overline{\boldsymbol{w}}) &= (4, 0, 5) \end{aligned}$$

Then, we obtain

$$\nabla f(\overline{\boldsymbol{w}})^T \Delta \boldsymbol{w} = (4, 0, 5)^T (1, 0, -2) = 4 \cdot 1 + 0 \cdot 0 + 5 \cdot (-2) = -6 < 0$$

Since the problem is minimization and $\nabla f(\overline{\boldsymbol{w}})^T \Delta \boldsymbol{w} < 0$, we conclude that $\Delta \boldsymbol{w}$ is an improving direction.

(b) The gradient is:

$$\begin{aligned} \nabla f(\boldsymbol{y}) &= \left(\frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial y_2} \right) = (9, 40) \\ \implies \nabla f(\overline{\boldsymbol{y}}) &= (9, 40) \end{aligned}$$

Observe that in this case the gradient is constant, that is, it does not depend on the point $\overline{\boldsymbol{y}}$. This happens when we have linear functions f.

Now we evaluate if the direction $\Delta \boldsymbol{y}$ is improving. We have:

$$\nabla f(\overline{\boldsymbol{y}})^T \Delta \boldsymbol{y} = (9, 40)^T (3, -6) = -213 < 0$$

Since the objective function is a maximization, $\Delta \boldsymbol{y}$ is not an improving direction.

(c) The gradient is:

$$\begin{aligned} \nabla f(\boldsymbol{z}) &= \left(\frac{\partial f}{\partial z_1}, \frac{\partial f}{\partial z_2} \right) = (10z_1 - 3z_2, \ -3z_1 + 2x_2) \\ \implies \nabla f(\overline{\boldsymbol{z}}) &= (1, 3) \end{aligned}$$

Then, we have

$$\nabla f(\overline{\boldsymbol{z}})^T \Delta \boldsymbol{z} = (1, 3)^T (-6, 2) = 0$$

so we need more information to draw a conclusion.

□

As stated earlier, the gradient shows the direction of most rapid value increase. But, can we use it as a move direction? The next principle shows us when it is a good idea.

Principle 3.7 (Principle 3.23 from the textbook)**.** *Whenever* $\nabla f(\boldsymbol{x}) \neq \boldsymbol{0}$,

- $\Delta \boldsymbol{x} = \nabla f(\boldsymbol{x})$ *is an improving direction for a maximize objective* f*, and*
- $\Delta \boldsymbol{x} = -\nabla f(\boldsymbol{x})$ *is an improving direction for a minimize objective* f.

Now we know how to compute improve directions, but how do we know if they are feasible? Recall our Example 3.4. The example illustrates that the feasible directions are different depending on the location of the current point in the feasible region. If our current point is in the interior (it is not touching any boundary point), then all directions are feasible; and if it is touching a boundary point, the number of feasible directions decreases. Let's formalize this idea with the following definition and principle.

Definition 3.8. *In an optimization problem, a constraint of the type* $g(\boldsymbol{x}) \leq b$ *is active at the solution* $\overline{\boldsymbol{x}}$ *if* $g(\overline{\boldsymbol{x}}) = b$.

That is, an inequality constraint is active at a solution $\overline{\boldsymbol{x}}$ if $\overline{\boldsymbol{x}}$ satisfies the constraint at equality.

Principle 3.8 (Principle 3.24 from the textbook)**.** *Whether a direction is feasible at a solution $\overline{\boldsymbol{x}}$ depends on whether it would lead to immediate violation of any active constraint at $\overline{\boldsymbol{x}}$.*

The methods used to decide whether a direction is feasible at a solution $\overline{\boldsymbol{x}}$ highly depend on the properties of the active constraints. In this course, we focus on linear problems, so let's develop conditions on linear constraints.

Recall that a linear constraint is a weighted sum of the variables. If we use $\boldsymbol{a}$ to denote the vector of weights and b to denote the right-hand side value, a linear constraint can be of the following three forms:

$$\boldsymbol{a}^T\boldsymbol{x} \geq b, \quad \boldsymbol{a}^T\boldsymbol{x} \leq b, \quad \boldsymbol{a}^T\boldsymbol{x} = b$$

In the following principle we establish conditions for feasible directions.

Principle 3.9 (Principle 3.25 from the textbook)**.** *The direction $\Delta\boldsymbol{x} = (\Delta x_1, \ldots, \Delta x_n)$ is feasible for a linearly constrained optimization model at the solution $\overline{\boldsymbol{x}} = (\overline{x}_1, \ldots, \overline{x}_n)$ if and only if the following three conditions are satisfied:*

(i) $\boldsymbol{a}^T\Delta\boldsymbol{x} \geq 0$ *for all active constraints of the form* $\boldsymbol{a}^T\overline{\boldsymbol{x}} \geq b$

(ii) $\boldsymbol{a}^T\Delta\boldsymbol{x} \leq 0$ *for all active constraints of the form* $\boldsymbol{a}^T\overline{\boldsymbol{x}} \leq b$

(iii) $\boldsymbol{a}^T\Delta\boldsymbol{x} = 0$ *for all equality constraints of the form* $\boldsymbol{a}^T\overline{\boldsymbol{x}} = b$

Proof. Let's sketch the proof. We evaluate feasibility of the direction $\Delta\boldsymbol{x}$ at $\overline{\boldsymbol{x}}$ by analyzing the left-hand side of the constraint, as follows:

$$\begin{aligned}\boldsymbol{a}^T(\overline{\boldsymbol{x}} + \lambda\Delta\boldsymbol{x}) &= \boldsymbol{a}^T\overline{\boldsymbol{x}} + \lambda\boldsymbol{a}^T\Delta\overline{\boldsymbol{x}} && \text{(Expanding the product)}\\ &= b + \lambda\boldsymbol{a}^T\Delta\overline{\boldsymbol{x}} && \text{(because the constraint is active at } \overline{\boldsymbol{x}})\end{aligned}$$

Since the step size $\lambda > 0$, the sign of $\boldsymbol{a}^T\Delta\overline{\boldsymbol{x}}$ will determine if $\Delta\boldsymbol{x}$ is a feasible direction according to the rules established in the principle. □

Let's see an example.

Example 3.6 (Examples 3.12 and 3.13 from the textbook)**.** *Consider the following optimization problem with linear constraints:*

$$\begin{aligned}\min \quad & f(\boldsymbol{w}) && (2)\\ s.t. \quad & 3w_1 + w_3 \geq 26 && (3)\\ & 5w_1 - 2w_3 \leq 50 && (4)\\ & 2w_1 + w_2 + w_3 = 20 && (5)\\ & w_1 \geq 0 && (6)\\ & w_3 \geq 0 && (7)\end{aligned}$$

(a) Determine whether the direction $\Delta\boldsymbol{w} = (0, -1, 1)$ is feasible at the solution $\overline{\boldsymbol{w}} = (6, 0, 8)$

(b) State the conditions that must be satisfied for a direction $\Delta\boldsymbol{w}$ to be a feasible move direction at $\overline{\boldsymbol{w}} = (10, 0, 0)$

Solution.

(a) We first check which constraints are active, plugging in the vector $\overline{\boldsymbol{w}}$ in each of the constraints. We obtain:

- Constraint (3):

$$3\overline{w}_1 + \overline{w}_3 = 3 \cdot 6 + 8 = 26$$

  Since 26 equals the right-hand side of the constraint, it is active.

- Constraint (4):

$$5\overline{w}_1 - 2\overline{w}_3 = 5 \cdot 6 - 2 \cdot 8 = 14$$

  Since $14 < 50$, this constraint is not active.

- Constraint (5) is an equality constraint and, therefore, it is always active.
- Constraints (6) and (7) are easy to check. Observe that:

$$\overline{w}_1 = 6 > 0 \qquad \text{and} \qquad \overline{w}_3 = 8 > 0$$

  Hence, constraint (6) is not active, and constraint (7) is active.

Now we use Principle 3.9 in constraints (3), (5) and (7).

- The vector $\boldsymbol{a}$ in constraint (3) is $\boldsymbol{a} = (3, 0, 1)$ and this is a $\geq$ constraint. Hence, we check whether $\boldsymbol{a}^T \Delta \boldsymbol{w} \geq 0$. We obtain:

$$\boldsymbol{a}^T \Delta \boldsymbol{w} = (3, 0, 1)^T (0, -1, 1) = 1 > 0$$

- The vector $\boldsymbol{a}$ in constraint (5) is $\boldsymbol{a} = (2, 1, 1)$ and this is an $=$ constraint. Hence, we check whether $\boldsymbol{a}^T \Delta \boldsymbol{w} = 0$. We obtain:

$$\boldsymbol{a}^T \Delta \boldsymbol{w} = (2, 1, 1)^T (0, -1, 1) = 0$$

- The vector $\boldsymbol{a}$ in constraint (7) is $\boldsymbol{a} = (0, 1, 0)$ and this is an $\geq$ constraint. Hence, we check whether $\boldsymbol{a}^T \Delta \boldsymbol{w} > 0$. We obtain:

$$\boldsymbol{a}^T \Delta \boldsymbol{w} = (0, 1, 0)^T (0, -1, 1) = -1 < 0$$

Since constraint (7) does not satisfy the conditions from Principle 3.9, we conclude that $\Delta \boldsymbol{w}$ is not a feasible direction at $\overline{\boldsymbol{w}}$.

(b) Similarly to the previous part, we first check what are the active constraints:

- Constraint (3):

$$3\overline{w}_1 + \overline{w}_3 = 3 \cdot 10 + 0 = 30 > 26$$

  so it is not active.
- Constraint (4):

$$5\overline{w}_1 - 2\overline{w}_3 = 5 \cdot 10 - 2 \cdot 0 = 50$$

  so it is active
- Constraint (5) is always active because it is an $=$
- Constraints (6) and (7):

$$\overline{w}_1 = 10 > 0 \qquad \text{and} \qquad \overline{w}_2 = 0$$

  so (6) is not active and (7) is active.

Now we use Principle 3.9 to write the conditions using the active constraints. We obtain:

$$\begin{aligned} &5\Delta w_1 - 2\Delta w_3 \leq 0 \qquad \text{(from constraint (4))} \\ &2\Delta w_1 + \Delta w_2 + \Delta w_3 = 0 \qquad \text{(from constraint (5))} \\ &\Delta w_2 \geq 0 \qquad \text{(from constraint (7))} \end{aligned}$$

□

4 Linear Programming Models

[This section is based on Chapter 4 of the textbook]

Linear problems are tremendously important in operations research because they are very easy to solve and a huge number of real-life applications can be modeled as linear programs.

In this section we will discuss some of the classical linear-programming formulations. Most of this models arose from real-life applications, and they have become popular due to their extensive applicability. In each case, we discuss the main characteristics of the type of problems and we solve an example.

4.1 Allocation models

The main issue in allocation models is to divide a limited resource among competing candidates.

Example 4.1 (Example 4.1 from the textbook)**.** *Bev is taking courses in operations research, economics, statistics and material science. She has 30 hours to prepare for her exams, and she wants to get the best possible performance in each class. Her favorite class is operations research, so she will spend as much on it as in any other subject. Still, she believes that 10 hours of study should be sufficient for each of the courses. She knows that each study hour on operations research increases her grade in 2%, each on economics 3%, each on statistics 1% and each on materials science 5%.*

Formulate an allocation problem to maximize her grades.

Solution. The decision variables are the amount of time to spend in each class. Then, for each $j \in \{1, 2, 3, 4\}$ define

$$x_j = \text{ hours to spend in subject } j,$$

where $j = 1$ is operations research, $j = 2$ economics, $j = 3$ statistics and $j = 4$ materials science.

Then, we obtain the following model

$$\begin{aligned}
\max \quad & 2x_1 + 3x_2 + x_3 + 5x_4 \\
\text{s.t.} \quad & \sum_{j=1}^{4} x_j = 30 \qquad \text{(allocation)} \\
& x_1 \geq x_j \qquad \forall j \in \{2, 3, 4\} \qquad \text{(study more OR)} \\
& x_j \leq 10 \qquad \forall j \qquad \text{(10 hours is enough)} \\
& x_j \geq 0 \qquad \forall j
\end{aligned}$$

□

The most characteristic constraint in this type of problems is the allocation constraint, which establishes that the sum of the allocated resources to the different destinations must be exactly (or at most in some cases) the total amount of resources that we have available.

The rest of the constraints are specific of each problem. There will usually be a "minimum performance" constraint, but it may come in different shapes. Let's see another example.

Example 4.2 (Exercise 4-1 from the textbook)**.** *Bisco's sugar-free, fat-free chocolate squares are so popular that the company cannot keep up with demand. Regional demands shown in the following table total 2000 cases per week, but Bisco can produce only 60% of that number.*

The table also shows the different profit levels per unit experienced in the regions due to competition and customer tastes. Bisco wants to find a maximum profit plan that fulfills between 50 and 70% of each region's demand.

	NE	*SE*	*MW*	*W*
Demand	*620*	*490*	*510*	*380*
Profit	*1.60*	*1.40*	*1.90*	*1.20*

Formulate an LP to choose an optimal distribution plan.

Solution. The decision variables are the how many chocolate squares to sell to each region. Then, we define

$$x_j = \text{ number of units to sell to region } j \qquad \forall j \in \{1,2,3,4\}$$

where $j = 1$ means NE, $j = 2$ means SE, $j = 3$ means MW, and $j = 4$ means W.

Then, we obtain the following model:

$$\begin{aligned}
\max \quad & 1.60x_1 + 1.40x_2 + 1.90x_3 + 1.20x_4 \\
\text{s.t.} \quad & x_1 + x_2 + x_3 + x_4 = 0.6 * 2000 \qquad \text{(allocation)} \\
& 0.5 * 620 \leq x_1 \leq 0.7 * 620 \qquad \text{(between 50 and 70\% of NE demand)} \\
& 0.5 * 490 \leq x_2 \leq 0.7 * 490 \qquad \text{(between 50 and 70\% of SE demand)} \\
& 0.5 * 510 \leq x_3 \leq 0.7 * 510 \qquad \text{(between 50 and 70\% of MW demand)} \\
& 0.5 * 380 \leq x_4 \leq 0.7 * 380 \qquad \text{(between 50 and 70\% of W demand)} \\
& x_j \geq 0 \qquad \forall j \in \{1,2,3,4\}
\end{aligned}$$

Observe that in this case the allocation constraint says that we must use all the units produced, so it can also be thought of as the availability of resources we have to sell.

Equivalently, we can define additional input parameters to obtain a more compact formulation. Specifically, for each $j \in \{1,2,3,4\}$, we define the input parameters:

$$\begin{aligned}
d_j &= \text{ demand in region } j \\
p_j &= \text{ profit in region } j
\end{aligned}$$

Then, we obtain the following model:

$$\begin{aligned}
\max \quad & \sum_{j=1}^{4} p_j x_j \\
\text{s.t.} \quad & 0.5 * d_j \leq x_j \leq 0.7 * d_j \qquad \forall j \in \{1,2,3,4\} \\
& x_j \geq 0 \qquad \forall j \in \{1,2,3,4\}
\end{aligned}$$

□

4.2 Blending models

In allocation problems, we take a single product and we try to split it to give it the best possible use. In blending models we do the opposite. We have a bunch of ingredients, and we try to mix them to create the best final product. Then, the decision is usually the amount of each ingredient to add to the final product so that it satisfies some sort of quality constraint.

In this section we will focus on two types of quality constraints that are very common. For each type of constraint we solve a small example.

Composition constraints

We start with an example and then we summarize the main ideas.

Example 4.3. *[Example 4.2 from the textbook] We have 3 ingredients available, and they have the following composition:*

- *Ingredient 1 is 4% fiber and has 10 mg of sodium per gram*
- *Ingredient 2 is 9% fiber and has 15 mg of sodium per gram*
- *Ingredient 3 is 3% fiber and has 5 mg of sodium per gram*

The goal is to cook a meal with these ingredients, satisfying the following constraints:

- *The blend must contain at most 100 mg of sodium*

- *The blend must have at least 5% fiber*

Solution. There is no clear objective function, so let's just write the constraints.
We first define the decision variables. For each $j \in \{1, 2, 3\}$, define

$$x_j = \text{ grams of ingredient } j \text{ to use}$$

Then, the following constraints must be satisfied:

$$\begin{aligned} &10x_1 + 15x_2 + 5x_3 \leq 100 && \text{(sodium)} \\ &0.04x_1 + 0.09x_2 + 0.03x_3 \geq 0.05 \sum_{i=1}^{3} x_i && \text{(fiber)} \\ &x_j \geq 0 \qquad \forall j \in \{1, 2, 3\} \end{aligned}$$

□

Observe that in both cases, the left-hand side of the constraint represents the total amount of the ingredient of interest in the final blend, and the right-hand side indicates the minimum/maximum amount allowed. For the right hand side, we showed two formats. In the sodium constraint, the right-hand side is an absolute constant (100). In the fiber constraint, the amount allowed is a percentage of the final product, and, hence, we multiply by the total amount of blend.

In general, these constraints follow the format:

$$\text{Total amount of ingredient } j \text{ in the total blend} \quad [\geq, \leq, =] \quad \text{Requirement}$$

Let's generalize a bit the example to practice this formulation, and the $\sum$, $\forall$ notation.

Example 4.4. *Let's generalize the model to n ingredients and m components. In other words, ingredient j has c_{ij} mg per gram of component i. Then, we want to write the following constraints:*

- *The blend must contain at most u_i mg of each component i*
- *The blend must have at least p_i% of each component i*

Solution. We first define our decision variables. For each $j \in \{1, \ldots, n\}$ define

$$x_j = \text{ grams of ingredient } j \text{ to put in the blend}$$

To write the constraint, observe that "the blend" refers to the total amount of ingredients we are putting together. Then, we obtain the following constraints:

$$\begin{aligned} &\sum_{j=1}^{n} c_{ij} x_j \leq u_i \qquad \forall i \in \{1, \ldots, m\} \\ &\frac{\sum_{j=1}^{n} c_{ij} x_j}{\sum_{j=1}^{n} x_j} \geq \frac{p_i}{100} \qquad \forall i \in \{1, \ldots, m\} \\ &x_j \geq 0 \qquad \forall j \in \{1, \ldots, n\} \end{aligned}$$

The second constraint is nonlinear, but since the denominator is nonnegative, we linearize it as follows:

$$\sum_{j=1}^{n} c_{ij} x_j \geq \frac{p_i}{100} \left(\sum_{j=1}^{n} x_j \right) \qquad \forall i \in \{1, \ldots, m\}$$

□

Ratio constraints

Many blending models have constraints of the form "the ratio between the first two ingredients must be at least 3/4", which we would formulate as

$$\frac{x_1}{x_2} \leq \frac{3}{4}.$$

This is clearly a nonlinear constraint, but we can reformulate it as a linear constraint, simply multiplying everything by the denominator on the left-hand side (provided that it's nonnegative). We obtain

$$x_1 \leq \frac{3}{4}x_2,$$

which is a linear constraint. If the denominator is not clearly positive, we cannot linearize ratio constraints.

Observe that the fiber constraint in Example 4.3 is a ratio constraint.

Example 4.5 (Example 4.3 from the textbook)**.** *Formulate linear constraints enforcing each of the following ratio requirements on nonnegative decision variables x_i that represent the amount of substance i in the final blend for $i \in \{1, 2, 3\}$.*

(i) The amounts of substance 1 and 2 should be in the ratio 4:7

(ii) The amount of substance 1 is at most half of that of substance 3

(iii) The blend is at least 40% substance 1

Solution. For each case, we write the constraint literally and, if needed, we linearize.

(a) The constraint is

$$\frac{x_1}{x_2} = \frac{4}{7},$$

which is nonlinear. To linearize, we multiply everything by x_2. Notice that, since the constraint is an $=$, we do not require $x_2 \geq 0$ to linearize. We obtain

$$x_1 = \frac{4}{7}x_2$$

(b) In this case, we immediately have the linear constraint

$$x_1 \leq \frac{1}{2}x_3$$

(c) In this case the constraint is saying that the proportion of the substance 1 with respect to the total blend has to be at least 40%. Then, the constraint is

$$\frac{x_1}{\sum_{i=1}^{3} x_i} \geq 0.4,$$

which we linearize as

$$x_1 \geq 0.4 \sum_{i=1}^{3} x_i.$$

□

4.3 Planning models

In planning problems, we are trying to decide what, when and where to do a series of activities. An example is the Pi Hybrids optimization model, where we had to decide how many bags of hybrids to produce in each facility, and where to ship the production. The main constraints were the capacity of the facilities and demand satisfaction.

A characteristic of planning models are the balance constraints. We will usually have two or more sets of decision variables that are related, and we need to include constraints that model this relationship. For example, in the Pi Hybrids example, we had a decision variable to decide how much of each hybrid to produce in each facility, and a decision variable to decide how much of each type of hybrid should be shipped from each facility to each sales region. Then, the balance constraint said that, for each hybrid, the total product shipped from each facility needs to be equal to the total production. Below we formally define a balance constraint.

Definition 4.1 (Def. 4.7 from the textbook)**.** *A balance constraint assures that in-flows equal or exceed out-flows of materials and products created by one stage of production and consumed by others.*

Example 4.6 (Example 4.4 from the textbook)**.** *Consider a manufacturer that assembles two products and uses two additional parts. Assembly 1 joins a product 2 with two part 3's and a part 4; and Assembly 2 joints one part 3 and two part 4's. Let x_i denote the number of parts or products manufactured, where $i = 1, 2$ correspond to the assemblies and $i = 3, 4$ to parts. Write the balance constraints.*

Solution. To be able to produce x_1 product 1, and x_2 product 2, we need to ensure that we have at least as many parts as required by the problem. Then, we have

$$\begin{aligned} x_2 &\geq x_1 && \text{(need of product 2)} \\ x_3 &\geq 2x_1 + x_2 && \text{(need of part 3)} \\ x_4 &\geq x_1 + 2x_2 && \text{(need of part 4)} \end{aligned}$$

□

Let's consider another example.

Example 4.7 (Example 4.5 from the textbook)**.** *An orange juice company can sell up to 15 thousand tons of juice to wholesalers at \$1500 per ton. The juice is either squeezed from oranges purchased at \$200 per ton, or diluted from concentrate obtained at \$1600 per ton. Approximately 10 thousand tons of oranges are available, and each ton yields 0.2 ton of juice. The supply of concentrate is unlimited, and each ton dilutes into 2 tons of juice.*

Formulate a linear program to choose an operating plan that maximizes the company's net income.

Solution. Let's define our decision variables:

$$\begin{aligned} x_1 &= \text{ Tons of oranges squeezed} \\ x_2 &= \text{ Tons of concentrate diluted for juice} \\ y &= \text{ Tons of juice sold} \end{aligned}$$

We can summarize the variables x as follows. For $i \in \{1, 2\}$, x_i represents the amount of source ingredient i to be transformed in juice, where $i = 1$ represents oranges and $i = 2$, concentrate.

The objective function is the net income, i.e., sales minus cost. We obtain

$$\max \quad 1500y - (200x_1 + 1600x_2)$$

We have three sets of constraints:

- Capacity constraints: There is limited amount of oranges and the demand is upper bounded too. Then, we obtain

$$\begin{aligned} x_1 &\leq 10,000 && \text{(availability of oranges)} \\ y &\leq 15,000 && \text{(max demand)} \end{aligned}$$

- Balance constraint: The amount of juice we sell should not exceed the production. Since each ton of oranges yields 0.2 tons of juice, and each ton of concentrate yields 2 tons of juice, the total production is $0.2x_1 + 2x_2$. Then, the balance constraint is

$$0.2x_1 + 2x_2 \geq y$$

- Nonnegativity:

$$x_1, x_2, y \geq 0$$

Hence, we obtain the following optimization problem

$$\begin{aligned} \max \quad & 1500y - (200x_1 + 1600x_2) \\ \text{s.t.} \quad & x_1 \leq 10,000 \qquad \text{(availability of oranges)} \\ & y \leq 15,000 \qquad \text{(max demand)} \\ & 0.2x_1 + 2x_2 \geq y \qquad \text{(balance)} \\ & x_1, x_2, y \geq 0 \end{aligned}$$

□

4.4 Shift scheduling and staff planning models

Shift scheduling can be considered opposite to planning models, similarly to blending models were opposite to allocation models. In planning models, we formulated an LP to decide how to use the available resources in the most efficient way. In shift scheduling and staff planning models, instead, we must decide how to use the resources to complete a pre-determined task in the best way.

Let's work on an example.

Example 4.8 (Example 4.6 from textbook)**.** *Clerical employees of a government agency are allowed to work four 10-hour days per week in any of the following patterns*

Pattern number	*Work days*
1	*Monday, Wednesday, Thursday, Friday*
2	*Monday, Tuesday, Thursday, Friday*
3	*Monday, Tuesday, Wednesday, Friday*

Formulate a linear program to determine the minimum number of employees needed to have at least 10 on duty Mondays, 9 on Fridays and 7 working on Tuesday through Thursdays.

Solution. We want to minimize the total number of employees we hire, subject to constraints depending on the pattern of their shifts. Then, our decision variables are

$$x_j \quad = \quad \text{Number of employees hired under pattern } j$$

for $j \in \{1, 2, 3\}$. Then, the objective function becomes:

$$\min \quad \sum_{j=1}^{3} x_j$$

The constraints are the number of people we need on duty on determined days. Specifically, we need:

- At least 10 workers on duty on Monday. Since all the patterns require the workers to be in the office on Monday, we have the following constraint:

$$x_1 + x_2 + x_3 \geq 10 \tag{8}$$

- At least 9 workers on duty on Friday. All the patterns require the workers in the office on Friday, so we obtain a similar constraint:

$$x_1 + x_2 + x_3 \geq 9 \tag{9}$$

Observe that constraint (8) is more restrictive than constraint (9). In other words, if (8) is satisfied, then (9) will be automatically satisfied. Then, we say that (9) is redundant and we do not need to write it in our LP.

- At least 7 workers on Tuesday through Thursday. In this case we write three constraints: one for Tuesday, one for Wednesday and one for Thursday. Observe that Tuesday is a work day only for patterns 2 and 3. Then, the constraint for Tuesday is

$$x_2 + x_3 \geq 7$$

Similarly, for Wednesday and Thursday we obtain

$$\begin{aligned} x_1 + x_3 &\geq 7 \qquad \text{(Wednesday requirement)} \\ x_1 + x_2 &\geq 7 \qquad \text{(Thursday requirement)} \end{aligned}$$

Putting everything together, we obtain the following optimization problem

$$\begin{aligned} \min \quad & \sum_{j=1}^{3} x_j \\ \text{s.t.} \quad & x_1 + x_2 + x_3 \geq 10 \qquad \text{(Monday requirement)} \\ & x_1 + x_2 + x_3 \geq 9 \qquad \text{(Friday requirement)} \\ & x_2 + x_3 \geq 7 \qquad \text{(Tuesday requirement)} \\ & x_1 + x_3 \geq 7 \qquad \text{(Wednesday requirement)} \\ & x_1 + x_2 \geq 7 \qquad \text{(Thursday requirement)} \\ & x_j \geq 0 \qquad \forall j \end{aligned}$$

As discussed earlier, we may omit the constraint for Friday and we would have an equivalent formulation. □
Observe that all the constraints in the example are of the form:

$$\sum_{\text{types of shifts}} \text{Work done in each shift} \geq \text{Minimum requirement}$$

This is known as a covering constraint, and it is characteristic of shift planning problems. Let's end this section with another example, which is a smaller version of Application 4.5 from the textbook.

Example 4.9. *The Ohio National Bank (ONB) is confronting a problem in staffing its check-processing center. Checks arrive through the morning in volumes peaking passed noon. Our fictitious version will assume the following arrivals (in thousands):*

Hour	*Arrivals (in thousands)*
9 am	*10*
10 am	*11*
11 am	*15*
12 pm	*20*
1 pm	*25*

Uncollected checks cost the bank money in lost interest. Thus, it is essential that all checks be processed in time for collection on the next business day. ONB decided to enforce a requirement that all checks be completed by 3 pm. Furthermore, the number of unprocessed at any hour should not exceed 20 thousand.

Two types of employees can perform the check processing task. Full-time employees process checks in 4-hour shifts with a break of one hour in the middle. Part-time employees process checks for 2 hours per day and do not have a break. Both types of shifts can start at any hour of the day, and full-time employees can be assigned an hour of overtime.

Full-time employees receive $11 per hour in pay and benefits, plus an extra $1 per hour for time after 1 pm and 150% pay for daily overtime. Part-time employees are paid $7 per hour, plus $1 per hour after 1 pm, and cannot work overtime. The possible shifts are presented in the next table, where R represents regular duty, O, possible overtime, and E, the extra money for working after 1 pm.

Time of the day	*Start of full-time shift*		*Start of part-time shift*				
	9 am	*10 am*	*9 am*	*10 am*	*11 am*	*12 pm*	*1 pm*
9 am	*R*		*R*				
10 am	*R*	*R*	*R*	*R*			
11 am		*R*		*R*	*R*		
12 pm	*R*				*R*	*R*	
1 pm (13 hrs)	*R+E*	*R+E*				*R+E*	*R+E*
2 pm (14 hrs)	*O+E*	*R+E*					*R+E*

Full-time employees work faster than part-time employees, with the rates of checks per hour being 1000 and 800, respectively.

One final complication is encoding stations. The number of machines available limits the number of employees who can work at any one time. Our center has 35 machines.

Formulate a linear program to minimize cost.

Solution. We will index variables by h, the hour at which a shift starts, where $h \in \{9, 10, 11, 12, 13, 14\}$.
To establish the decision variables, observe that we actually have 3 types of workers:

(i) Full-time employees that do not work overtime

(ii) Full-time employees that work overtime

(iii) Part-time employees

Then, we assign a symbol to each of them. We obtain the following decision variables

x_h = Number of full-time employees who do not work overtime, starting shift at time h ($h \in \{9, 10\}$)
y_h = Number of full-time employees who work overtime, starting shift at time h ($h \in \{9\}$)
z_h = Number of part-time employees starting shift at time h ($h \in \{9, 10, 11, 12, 13\}$)

To compute the objective function, we compute the cost of each shift. We need to consider the overtime and number of hours worked after 1 pm. We obtain the following table

Length of shift	Hours overtime	Hours after 1 pm	Total cost
Full time (4 hours)	1	1 regular + 1 overtime	$3 * 11 + 12 + 12 * 1.5 = 63$
Full time (4 hours)	0	1	$3 * 11 + 12 = 45$
Full time (4 hours)	0	2	$2 * 11 + 2 * 12 = 46$
Part time (2 hours)	0	0	$2 * 7 = 14$
Part time (2 hours)	0	1	$7 + 8 = 15$
Part time (2 hours)	0	2	$2 * 8 = \$16$

Then, the objective function is:

$$\min \quad 63y_9 + 45x_9 + 46x_{10} + 14(z_9 + z_{10} + z_{11}) + 15z_{12} + 16z_{13}$$

Now we write the constraints.

- Availability of machines: Using the table of shifts, we obtain

$$\begin{aligned}
&x_9 + y_9 + z_9 \leq 35 && \text{(machines at 9 am)}\\
&x_9 + y_9 + x_{10} + z_9 + z_{10} \leq 35 && \text{(machines at 10 am)}\\
&x_{10} + z_{10} + z_{11} \leq 35 && \text{(machines at 11 am)}\\
&x_9 + y_9 + z_{11} + z_{12} \leq 35 && \text{(machines at 12 pm)}\\
&x_9 + y_9 + x_{10} + z_{12} + z_{13} \leq 35 && \text{(machines at 1 pm)}\\
&y_9 + x_{10} + z_{13} \leq 35 && \text{(machines at 2 pm)}
\end{aligned}$$

- Covering constraints: We need to finish processing all the checks by 3 pm, and in each hour we cannot be backlogged in more than 20 thousand checks. To write these constraints we need an additional decision variable. Define:

  w_h = Number of checks that have arrived and were not processed in hour $h-1$ (in thousands) ($h \in \{10, 11, 12, 13, 14\}$)

  Technically, we should define w_{15} too, but since we cannot leave any checks unprocessed, we would have a constraint $w_{15} = 0$. Both ways to write the LP are correct.

  Then, the covering constraints are:

$$
\begin{aligned}
& x_9 + y_9 + 0.8z_9 \geq 10 - w_{10} && \text{(9 am covering)} \\
& x_9 + y_9 + x_{10} + 0.8z_9 + 0.8z_{10} \geq 11 + w_{10} - w_{11} && \text{(10 am covering)} \\
& x_{10} + 0.8z_{10} + 0.8z_{11} \geq 15 + w_{11} - w_{12} && \text{(11 am covering)} \\
& x_9 + y_9 + 0.8z_{11} + 0.8z_{12} \geq 20 + w_{12} - w_{13} && \text{(12 pm covering)} \\
& x_9 + y_9 + x_{10} + 0.8z_{12} + 0.8z_{13} \geq 25 + w_{13} - w_{14} && \text{(1 pm covering)} \\
& y_9 + x_{10} + 0.8z_{13} \geq w_{14} && \text{(2 pm covering)}
\end{aligned}
$$

- The number of unprocessed at any hour should not exceed 20 thousand:

$$w_h \leq 20 \qquad \forall h \in \{10, 11, 12, 13, 14\}$$

- Nonnegativity:

$$
\begin{aligned}
& x_9, x_{10}, y_9 \geq 0 \\
& z_h \geq 0 \qquad \forall h \in \{9, 10, 11, 12, 13\} \\
& w_h \geq 0 \qquad \forall h \in \{10, 11, 12, 13, 14\}
\end{aligned}
$$

For brevity, we don't write the entire optimization problem. □

In the examples of this section we build linear programs using variables that should be integers. However, solving a problem with integer decision variables is super hard compared to continuous variables. Then, we have to evaluate if the extra effort is worth it. If we expect our solutions to be in the order of 20's of more, then we can approximate to the fractional solution to an integer number and we would not change the problem a lot. However, if our solutions' magnitude is below 10, then fractional solutions can make huge differences. More formally, we have the following principle.

Principle 4.1 (Principle 4.6 from the textbook)**.** *To gain tractability with little loss of validity, decision variables of relatively large magnitude are best modeled as continuous, even though they correspond to physically integer quantities.*

In the banking example, we modeled the problem in an hour-per-hour fashion to count the number of workers that can process checks. However, in the covering constraints we saw that some of the work of the current hour can be passed to the next hour. We will see more models like this in the next subsection.

4.5 Time-phased models

So far, most of the problems we have worked on are *static*, which means that we solve the problem for a single day. However, in real-life applications we need to be able to model *dynamic* problems, i.e., problems that evolve in time.

Let's work on an example.

Example 4.10 (Exercise 4-16 from the textbook)**.** *The Big Gear (BG) transmission company buys and distributes replacement transmissions for large, 18-wheeler trucks. For the next 4 months, the company anticipates demands of 100, 130, 95 and 300 units, respectively. During the first month, units can be purchased from BG's supplier at a cost of \$12K each, but thereafter the price goes to \$14K per transmission. One order will be placed at the beginning of each month, and goods arrive immediately. Units can be held in the BG warehouse at a cost of \$1.2K per unit held per month, and there is no starting inventory. Naturally, the company wants to meet demand over the finite 4-month horizon at minimum total cost.*

Assuming that the demand occurs on the last day of the month, formulate a linear program to minimize cost.

Solution. Let's write a generic formulation of this problem. Let T be the set of months for which we want to optimize, and define the following input parameters. For each $t \in T$, define:

$$\begin{array}{rcl} c_t & = & \text{Cost of purchasing an item in month } t \\ s_t & = & \text{Cost of storing one unit in month } t \\ d_t & = & \text{Demand in month } t \end{array}$$

Observe that, according to the data we have, $T = \{1, 2, 3, 4\}$, the purchasing costs are $c_1 = 12$ and $c_2 = c_3 = c_4 = 14$, storing costs are $s_1 = s_2 = s_3 = s_4 = 1.2$ and the demand, $d_1 = 100$, $d_2 = 130$, $d_3 = 95$ and $d_4 = 300$.

Now let's formulate the problem as a linear program. We are interested in minimizing cost, so to decide our decision variables we look for the sources of costs. In this case, we can spend money purchasing items each month and storing them from one month to next. Then, our decision variables are:

$$\begin{array}{rcl} x_t & = & \text{Units purchased in month } t \ (t \in T) \\ h_t & = & \text{Units stored in month } (t \in T) \end{array}$$

Then, the objective function is to minimize the total cost, i.e., the cost of purchasing items and the cost of storing items. We obtain

$$\min \quad \sum_{t \in T} c_t x_t + \sum_{t \in T} s_t h_t$$

The only constraints are demand satisfaction, and balance constraints to determine the amount of storage in each month. However, demand satisfaction is a consequence of the balance constraints, as we will see below.

The idea behind the time-phased balance constraints is that we cannot have lost items. Then, what we have at the beginning of each month (purchases and inventory from previous month) must equal what we sell (demand) plus what we store for next month. Then, we obtain the following set of constraints:

$$\begin{aligned} & x_1 = d_1 + h_1 \\ & x_t + h_{t-1} = d_t + h_t \qquad \forall t > 1,\ t \in T \end{aligned}$$

Since there is no inventory at the beginning of the time period we study, we write a different constraint for the first month. An alternate approach would be to define an additional parameter for the initial inventory and set it to zero.

Observe that, by definition, $h_{t-1} \geq 0$. Therefore, our time-phased balance constraint immediately implies that $x_t + h_{t-1} \geq d_t$, that is, the demand satisfaction constraint.

Hence, including the nonnegativity constraints, we have the following optimization problem:

$$\begin{aligned} \min \quad & \sum_{t \in T} c_t x_t + \sum_{t \in T} s_t h_t \\ \text{s.t.} \quad & x_1 = d_1 + h_1 \\ & x_t + h_{t-1} = d_t + h_t \qquad \forall t > 1,\ t \in T \qquad \text{(time-phased balance)} \\ & x_t, h_t \geq 0 \qquad \forall t \in T \qquad \text{(nonnegativity)} \end{aligned}$$

□

The most characteristic constraint of time-phased models is the time-phased balance constraint. In the example, this constraint was in terms of the demand, production and inventory (as in many other cases), but it may include other variables. In general, we formulate the time-phased balance constraints as follows:

$$\text{start at period } t + \text{impact of period } t = \text{start of period } (t+1)$$

4.6 Models with Linearizable Nonlinear Objectives and Constraints

Most of the time, nonlinear problems cannot be written as equivalent linear problems. If that's the case, we may want to re-think our model or deal with the nonlinearities. In this course, we won't solve nonlinear problems. However, we will learn how to linearize some nonlinearities.

We describe them in the principle below, and we will do some examples.

Principle 4.2. *Assuming that $\boldsymbol{x}$ is the vector of decision variables and that $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ are linear functions, we can reformulate the following nonlinear objective functions as linear functions*

(i) $\min \max\{f_1(\boldsymbol{x}),\ f_2(\boldsymbol{x})\}$

(ii) $\max \min\{f_1(\boldsymbol{x}),\ f_2(\boldsymbol{x})\}$

In both cases, we introduce an additional decision variable y and obtain the following linear problems.

(i) In this case, the objective function is $f(\boldsymbol{x}) = \max\{f_1(\boldsymbol{x}),\ f_2(\boldsymbol{x})\}$ and, since max *is the biggest of the two functions, minimizing f is equivalent to minimizing f_1 and f_2 at the same time. Then, we reformulate the problem as follows*

$$\begin{aligned} \min \quad & y \\ s.t. \quad & f_1(\boldsymbol{x}) \leq y \\ & f_2(\boldsymbol{x}) \leq y \end{aligned}$$

(ii) Similarly, in this case the objective function is $f(\boldsymbol{x}) = \min\{f_1(\boldsymbol{x}), f_2(\boldsymbol{x})\}$ and, since min *is the smallest of the two functions, we reformulate as follows*

$$\begin{aligned} \max \quad & y \\ s.t. \quad & f_1(\boldsymbol{x}) \geq y \\ & f_2(\boldsymbol{x}) \geq y \end{aligned}$$

Principle 4.3. *The reformulation of constraints is similar. Suppose $g_1(\boldsymbol{x}), g_2(\boldsymbol{x})$ and $g_3(\boldsymbol{x})$ are linear functions of $\boldsymbol{x}$. Then, we can reformulate the following nonlinear constraints as linear constraints*

(i) $\max\{g_1(\boldsymbol{x}), g_2(\boldsymbol{x})\} \leq b$

(ii) $\min\{g_1(\boldsymbol{x}), g_2(\boldsymbol{x})\} \geq b$

(iii) $\max\{g_1(\boldsymbol{x}),\ g_2(\boldsymbol{x})\} + g_3(\boldsymbol{x}) \leq b$

(iv) $\min\{g_1(\boldsymbol{x}),\ g_2(\boldsymbol{x})\} + g_3(\boldsymbol{x}) \geq b$

To reformulate these constraints as linear constraints, we just separate them into two constraints as follows

(i) Observe that in this case the constraint is of the form $g(\boldsymbol{x}) \leq b$, where $g(\boldsymbol{x}) = \max\{g_1(\boldsymbol{x}), g_2(\boldsymbol{x})\}$. Then, we rewrite the constraint as follows:

$$g_1(\boldsymbol{x}) \leq b$$
$$g_2(\boldsymbol{x}) \leq b$$

(ii) Similarly, in this case we reformulate the constraint as follows

$$g_1(\boldsymbol{x}) \geq b$$
$$g_2(\boldsymbol{x}) \geq b$$

(iii) In this case, the constraint is of the form $g(\boldsymbol{x}) \leq b$, where $g(\boldsymbol{x}) = \max\{g_1(\boldsymbol{x}), g_2(\boldsymbol{x})\} + g_3(\boldsymbol{x})$. Since now the maximum is part of a sum, we treat it separately (similarly to the objective function case). We include a new decision variable z and we write

$$z + g_3(\boldsymbol{x}) \leq b$$
$$g_1(\boldsymbol{x}) \leq z$$
$$g_2(\boldsymbol{x}) \leq z$$

(iv) In this case we also include an additional decision variable z, and we reformulate as follows:

$$z + g_3(\boldsymbol{x}) \geq b$$
$$g_1(\boldsymbol{x}) \geq z$$
$$g_2(\boldsymbol{x}) \geq z$$

Observe that in the case of constraints, we can only give an equivalent linear reformulation if they are of the form "max $\leq b$" or "min $\geq b$". If the inequality sign is flipped, we cannot reformulate as linear constraints.

We will see examples below.

Example 4.11. *Assuming that x_1 and x_2 are decision variables, reformulate the following optimization problem as a linear problem. If it's not possible, explain why.*

$$\begin{aligned} \min \quad & \max\{2x_1 + x_2,\ x_1 + 2x_2\} \\ s.t. \quad & |x_1 + x_2| + \max\{x_1, x_2\} \leq 1 \end{aligned}$$

Solution. To reformulate the objective function, we use case (i) from Principle 4.2. Hence, we include the decision variable y and reformulate the objective function as follows:

$$\min \quad y$$

$$\begin{aligned} \text{s.t.} \quad & 2x_1 + x_2 \leq y \\ & x_1 + 2x_2 \leq y \end{aligned}$$

To reformulate the constraint, we first observe that $|a| = \max\{a, -a\}$. Then,

$$|x_1 + x_2| = \max\{x_1 + x_2, -x_1 - x_2\}.$$

Then, we use part (iii) of Principle 4.3 for $|x_1 + x_2|$ and for $\max\{x_1, x_2\}$. Including the decision variable z_1 for $|x_1 + x_2|$ and z_2 for $\max\{x_1, x_2\}$ we obtain

$$\begin{aligned} & z_1 + z_2 \leq 1 \\ & x_1 + x_2 \leq z_1 \\ & -x_1 - x_2 \leq z_1 \\ & x_1 \leq z_2 \\ & x_2 \leq z_2 \end{aligned}$$

Hence, the linear reformulation of the optimization problem is

$$\begin{aligned} \min \quad & y \\ \text{s.t.} \quad & 2x_1 + x_2 \leq y \\ & x_1 + 2x_2 \leq y \\ & z_1 + z_2 \leq 1 \\ & x_1 + x_2 \leq z_1 \\ & -x_1 - x_2 \leq z_1 \\ & x_1 \leq z_2 \\ & x_2 \leq z_2 \end{aligned}$$

□

The following example describes a realistic nonlinear cost function.

Example 4.12. *A farm grows two types of mangoes. This year the yield of mangoes of type 1 is 1000 units, and of type 2 is 700 units. The mangoes can be sold to a retail company and can also be sold in a local market directly. The contract with the retail company has the following details:*

(1) At least 60% of the total amount of mangoes sold to the retail company must be of type 2

(2) The retail company has a demand of 1000 units of mangoes and pays \$3 per unit of both types of mangoes

(3) It is possible to sell less than the demand to the retail company: If the total amount of mangoes sold to the retail company is less than 1000 units, then there is a penalty of \$0.05 per unit shortfall from 1000 units. For example, if the farm decides to sell 900 units, then they earn $900 \times \$3 - (1000 - 900) \times \$0.05$

(4) It is possible to sell more than the demand to the retail company. However, if the number of units of mangoes is more than 1200, then each unit over 1200 is purchased by the retail store at \$2.5 per unit. For example, if the farm decides to sell 1300 mangoes to the retail company, then the farm earns $1200 \times \$3 + (1300 - 1200) \times \$2.5$

Mangoes type 1 sell at \$2.2 per unit in the local market, and mangoes type 2 sell at \$3.1 per unit in the local market. Any amount of mangoes can be sold in the local market.

Formulate a linear program to maximize the net revenue of the farm.

Solution. We first define the decision variables. Since everything we don't sell to the retail store can be sold at the local market, we only need to track the number of mangoes that we sell to the retailer. Also, we need to track the type of mango we sell. Then, we define

$$x_i \quad = \quad \text{Number of mangoes type } i \text{ that we sell to the retail store } (i \in \{1, 2\})$$

Then, the number of mangoes type 1 we sell to the local market is $1000 - x_1$ and type 2, $700 - x_2$.

Before writing the model, let's work on the cost structure of the contract with the retail company. They pay \$3 per mango if we sell them between 1000 and 1200 mangoes. If we sell less or more than that, we pay penalties as follows:

- If we sell them less than 1000 mangoes, we pay \$0.05 per mango shortfall from 1000. Then, we pay a penalty u defined as follows:

$$u = \begin{cases} 0 & \text{if } x_1 + x_2 \geq 1000 \\ (1000 - x_1 - x_2) \cdot 0.05 & \text{if } x_1 + x_2 \leq 1000 \end{cases}$$

  and this cost structure can be rewritten as:

$$u = \max\{(1000 - x_1 - x_2) \cdot 0.05,\ 0\}$$

- If we sell more than 1200 mangoes to the retailer, we pay a penalty of \$0.5 per mango over 1200. Then we can define a penalty v as follows:

$$v = \begin{cases} (x_1 + x_2 - 1200) \cdot 0.5 & \text{if } x_1 + x_2 \geq 1200 \\ 0 & \text{if } x_1 + x_2 \leq 1200 \end{cases}$$

  and this penalty can also be described by the following function:

$$v = \max\{(x_1 + x_2 - 1200) \cdot 0.5,\ 0\}$$

Then, we have the following model:

$$\begin{aligned} \max \quad & 3(x_1 + x_2) + 2.2(1000 - x_1) + 3.1(700 - x_2) - \max\{(1000 - x_1 - x_2) \cdot 0.05,\ 0\} - \max\{(x_1 + x_2 - 1200) \cdot 0.5,\ 0\} \\ \text{s.t.} \quad & x_2 \geq 0.6(x_1 + x_2) \qquad \text{(condition (1))} \\ & x_1 \leq 1000 \qquad \text{(availability of mangoes type 1)} \\ & x_2 \leq 700 \qquad \text{(availability of mangoes type 2)} \\ & x_i \geq 0 \qquad \forall i \in \{1, 2\} \end{aligned}$$

Of course, this model is nonlinear because the green and Orangeorange expressions involve a max function. Since these max functions have a negative sign, we are under the structure of a $\max\min\{f_1, f_2\}$ function. To linearize, we define additional decision variables for these penalties. Let

$$\begin{aligned} u &= \text{Penalty incurred for selling less than 1000 units to the retail company} \\ v &= \text{Loss in revenue for selling more than 1200 units to the retail company} \end{aligned}$$

Then, we obtain the following linear formulation:

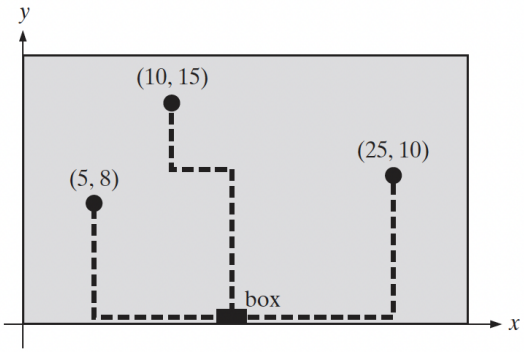
$$\begin{aligned} \max \quad & 3(x_1 + x_2) + 2.2(1000 - x_1) + 3.1(700 - x_2) - u - v \\ \text{s.t.} \quad & x_2 \geq 0.6(x_1 + x_2) \qquad \text{(condition (1))} \\ & u \geq 0.05(1000 - x_1 - x_2) \qquad \text{(condition (3))} \\ & u \geq 0 \\ & v \geq 0.5(x_1 + x_2 - 1200) \qquad \text{(condition (4))} \\ & v \geq 0 \\ & x_1 \leq 1000 \qquad \text{(availability of mangoes type 1)} \\ & x_2 \leq 700 \qquad \text{(availability of mangoes type 2)} \\ & x_i \geq 0 \qquad \forall i \in \{1, 2\} \end{aligned}$$

□

4.7 Examples to learn AMPL

In this section we work on two examples that we will solve in AMPL. We provide the model and a screenshot of the AMPL code.

Example 4.13. *The following figure shows the ceiling locations of 3 sensors in a new factory relative to a coordinate system (in feet) with origin at the lower left. A control box will be located along the long (lower in the figure) wall with fiber-optic cables running rectilinearly to each sensor. Designers want to place the box to minimize the cable required.*



Solution. The problem asks us to find the best location for the control box on the horizontal axis. Hence, there is only one decision variable:

$$x \quad : \quad \text{Location of the control box on the horizontal axis}$$

Then, the total cable required to connect the control box with a sensor in position (i, j) is:

$$|x - i| + j$$

Hence, we obtain the following (nonlinear) optimization problem:

$$\begin{array}{ll} \min & |x-5| + 8 + |x-10| + 15 + |x-25| + 10 \\ \text{s.t.} & x \geq 0 \end{array}$$

However, this is a nonlinear problem because there are absolute values in the objective function. Recall that

$$|z| = \max\{z, -z\}$$

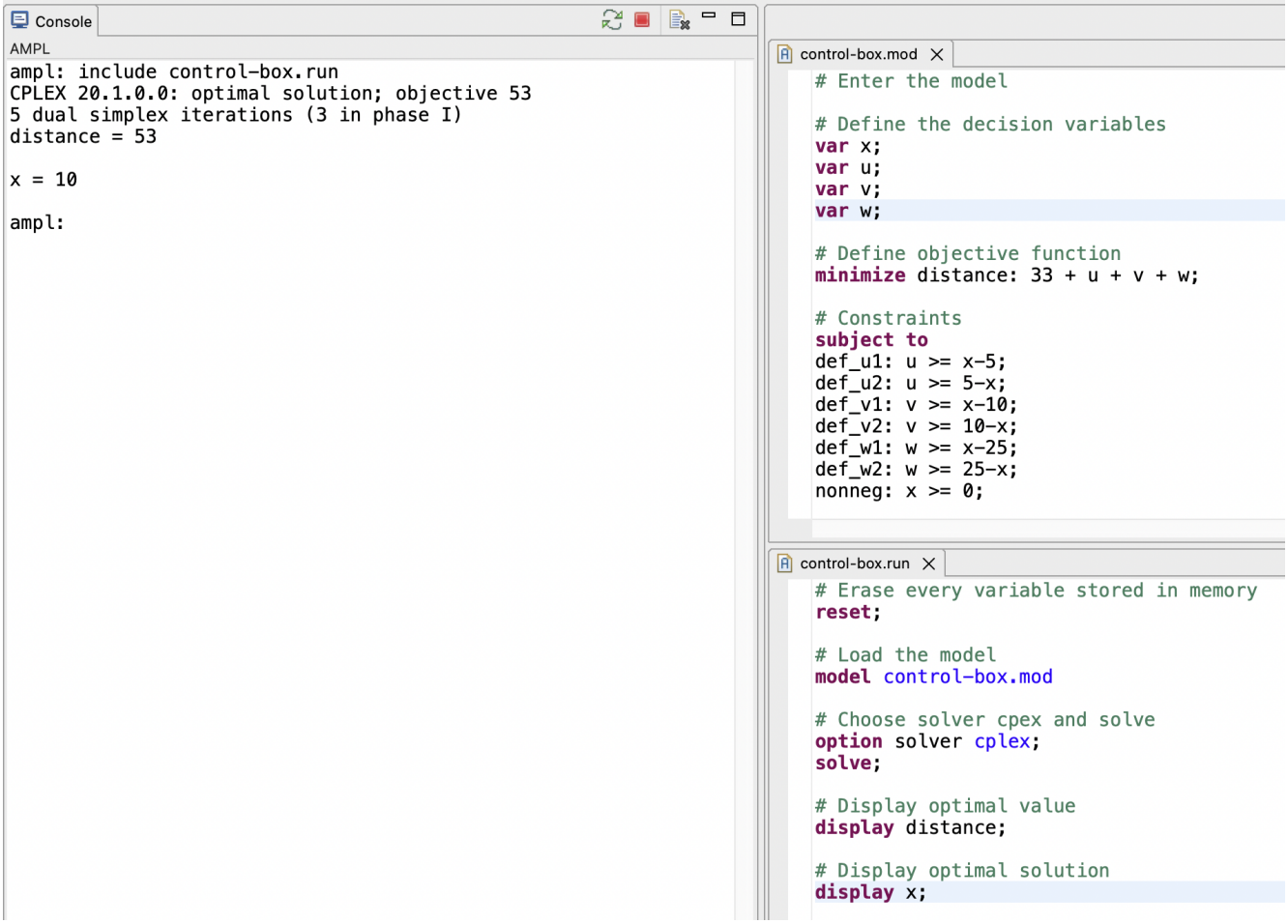
Then, we can linearize the objective function. To do that, we add the decision variables:

$$\begin{array}{lcl} u & : & \text{Distance between control box and sensor in position (5,8)} \\ v & : & \text{Distance between control box and sensor in position (10,15)} \\ w & : & \text{Distance between control box and sensor in position (25,10)} \end{array}$$

Hence, we obtain the following linear problem:

$$\begin{array}{ll} \min & 33 + u + v + w \\ \text{s.t.} & u \geq x - 5 \\ & u \geq 5 - x \\ & v \geq x - 10 \\ & v \geq 10 - x \\ & w \geq x - 25 \\ & w \geq 25 - x \\ & x \geq 0 \end{array}$$

Now we enter the model to AMPL. In the `.mod` file we enter the model, and in the `.run` file we write the instructions to run and solve the model. To run the model, we use the command `include` in the console. The code along with the output are presented in the following screenshot:



Then, the optimal solution is $x^* = 10$ and the total amount of cable required (optimal value) is 53 feet. Observe that the box is going to be placed right under the sensor in position $(10, 15)$. □

In the following example we include more data and, hence, we will need a data file in the AMPL code.

Example 4.14. *Cattle feed can be mixed from oats, corn, alfalfa, and peanut hulls. The table below shows the current cost per ton (in dollars) of each of these ingredients, together with the percentage of recommended daily allowances for protein, fat, and fiber that a serving of it fulfills.*

	Oat	*Corn*	*Alfalfa*	*Peanut hulls*
% protein	*60*	*80*	*55*	*40*
% fiber	*90*	*30*	*60*	*80*
% fat	*50*	*70*	*40*	*100*
Cost	*200*	*150*	*100*	*75*

We want to find a minimum cost way to produce feed that satisfies at least 60% of the daily allowance for protein and fiber while not exceeding 60% of the fat allowance.

Formulate a blending LP to choose an optimal feed mix.

Solution. To write a compact model, we introduce the following notation for the input parameters:

p_{ij} : Percentage of daily allowance of nutrient i in ingredient j
c_j : Cost of per ton of ingredient j

where $i \in \{1,2,3\}$ with $i = 1$ denoting protein, $i = 2$ fiber and $i = 3$ fat; and $j \in \{1,2,3,4\}$ with $j = 1$ denoting oat, $j = 2$ corn, $j = 3$ alfalfa and $j = 4$ peanut hulls. Next, we introduce the decision variables:

x_j : Proportion of ingredient j to include in the feed mix, $\forall j \in \{1,\ldots,4\}$.

We obtain the following linear model:

$$\begin{aligned}
\min \quad & \sum_{j=1}^{4} c_j x_j \\
\text{s.t.} \quad & \sum_{j=1}^{4} p_{ij} x_j \geq 60 \qquad \forall i \in \{1,2\} \qquad \text{(at least 60\% of protein and fiber)} \\
& \sum_{j=1}^{4} p_{3,j} x_j \leq 60 \qquad \text{(at most 60\% of fat)} \\
& \sum_{j=1}^{4} x_j = 1 \qquad (x_j \text{ represent proportions, so must add up to 1}) \\
& x_j \geq 0 \qquad \forall j \in \{1,2,3,4\}
\end{aligned}$$

The AMPL code is presented below. Observe that now we also use a `.dat` file to introduce the data.

Console

```
AMPL
ampl: include cat-food.run
CPLEX 20.1.0.0: optimal solution; objective 125
3 dual simplex iterations (0 in phase I)
cost = 125

x [*] :=
1  0.166667
2  0.166667
3  0.666667
4  0
;

ampl:
```

cat-food.mod

```
# Introduce notation for input parameters
param p {i in 1..3, j in 1..4};
param c {j in 1..4};

# Define decision variables
var x{j in 1..4};

# Objective function
minimize cost: sum{j in 1..4} c[j]*x[j];

# Constraints:
subject to
protein_fiber {i in 1..2}: sum{j in 1..4} p[i,j]*x[j] >= 60;
fat: sum{j in 1..4} p[3,j]*x[j] <= 60;
total: sum{j in 1..4} x[j] = 1;
nonneg {j in 1..4}: x[j] >= 0;
```

cat-food.run

```
# Erase all stored variables
reset;

# Load the model
model cat-food.mod

# Load the data
data cat-food.dat

# Load solver and solve
option solver cplex;
solve;

# Display optimal value and optimal solution
display cost, x;
```

cat-food.dat

```
# Here we enter the data
data;

# Below we enter the data, that is the value of each parameter
### use tab (not spaces) for indentation and separating numbers

# Define paramater h[j] for each j=1,2,3,4
param c:=
    1   200
    2   150
    3   100
    4   75  ;

# Define parameter s[i,j] for each i=1,2,3 and j=1,2,3,4
param p: 1  2   3   4:=
    1   60  80  55  40
    2   90  30  60  80
    3   50  70  40  100 ;
```

The optimal solution is 0.167 of oat, 0.167 of corn, 0.667 of alfalfa and no peanut hulls, and the optimal cost is \$125 per ton of food.

□

5 Simplex Algorithm

[This section is based on Chapter 5 of the textbook]

We have spent a lot of energy revising some of the most popular linear models, and linearizing some nonlinear objective functions and constraints. In this chapter we will exploit the properties of linear problems and learn the most popular algorithm to solve them: Simplex. We start with some geometry, that gives the foundations of why the linear programs are easy to solve and what makes Simplex so good.

5.1 LP optimal solutions and standard form

We first focus on the nice properties of the feasible set of any linear program. Let's call X this feasible set.

(i) The feasible set of a linear program is convex, i.e., for any pair of points $\boldsymbol{x}, \boldsymbol{y} \in X$, the line between them is part of the set. Formally, for any $\lambda \in [0,1]$, we have $\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y} \in X$. Intuitively, convexity means that there are no "holes" in X.

(ii) Since X is the intersection of linear inequalities, it is in fact a polyhedron.

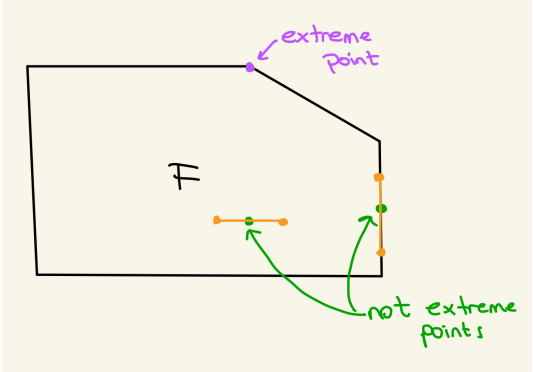
Convexity is important because it makes the search for an optimal solution easier, since the set does not have any "holes". But why is the polyhedral structure important? Let's study some properties of polyhedra.

Properties and definitions for polyhedra

In this subsection we suppose that F is a polyhedron.

Definition 5.1. *A point $\boldsymbol{x} \in F$ is an extreme point of F if it does not lie within the line segment between any 2 other members of F.*

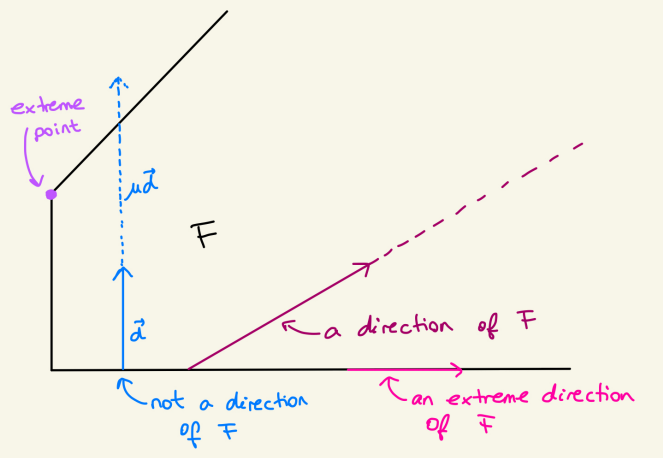
In the next figure we show a pictorial example of what is an extreme point and what is not. We show one extreme point and two points that are not extreme points. For the points that are not extreme, we show that they are part of the line between two other points of F.



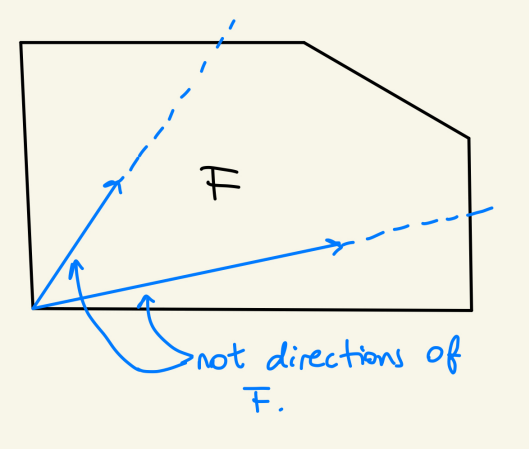
In simple words, the extreme points of a polyhedron are the vertices.

Definition 5.2. *A direction $\boldsymbol{d}$ is a direction of F if for some point $\boldsymbol{x} \in F$, the vector $\boldsymbol{x} + \mu \boldsymbol{d} \in F$ for all $\mu > 0$. Such $\boldsymbol{d}$ is an extreme direction of F if it cannot be written as a combination $\mu_1 \boldsymbol{d}_1 + \mu_2 \boldsymbol{d}_2$ of other directions $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ of F with $\mu_1 > 0$ and $\mu_2 > 0$.*

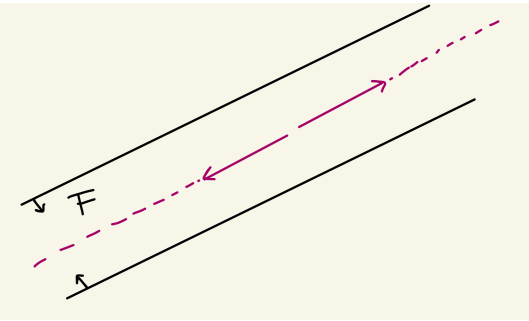
In the following picture we show an extreme point, a direction that is not an extreme direction, an extreme direction and a vector that is not a direction.



Observe that, since the definition of direction includes "any size of the vector", only unbounded polyhedra have directions. In the next picture we show that bounded polyhedra do not have directions.



We saw an example of a polyhedron that has directions, and one that does not. But do all polyhedra have extreme points? Not really. Only the ones that do not contain a line, as shown in the following picture.



Intuitively, a polyhedron has extreme points unless it is so loosely constrained that a direction can be pursued in both, positive and negative steps, without losing feasibility. This property is highly unlikely in practical problems because we usually have the nonnegativity constraints (at least).

A key property of polyhedra is that we can completely characterize a polyhedral set with their extreme points and extreme directions. In other words, having the set of inequalities that define them is exactly the same than knowing their extreme points and extreme directions. We formalize this idea below.

Theorem 5.1. *Suppose the polyhedron F has extreme points $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}$ and extreme directions $\boldsymbol{d}^{(1)}, \ldots, \boldsymbol{d}^{(k)}$.*

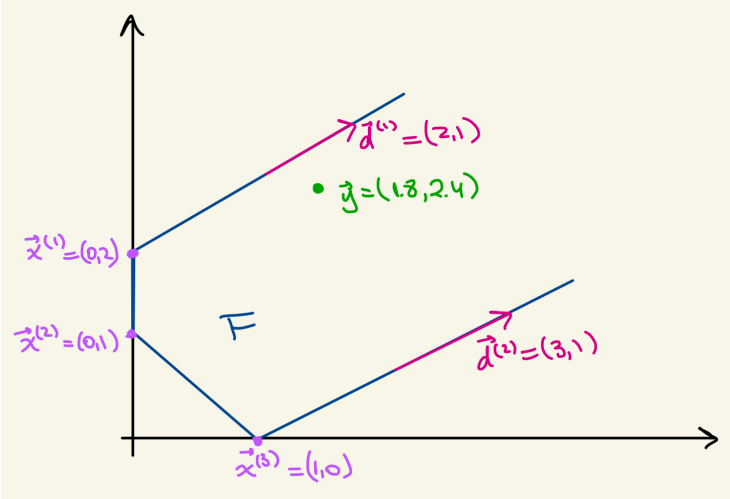
Then, a point $\boldsymbol{y}$ *belongs to* F *if and only if*

$$\boldsymbol{y} = \sum_{i=1}^{n} \lambda_i \boldsymbol{x}^{(n)} + \sum_{j=1}^{k} \mu_j \boldsymbol{d}^{(k)}$$

where $\lambda_i \geq 0$ *for all* $i \in \{1, \ldots, n\}$ *satisfy* $\sum_{i=1}^{n} \lambda_i = 1$ *and* $\mu_j \geq 0$ *for all* $j \in \{1, \ldots, k\}$.

In the example from the following picture, we have that the point $\boldsymbol{y} = (1.8, 2.4)$ can be written as

$$\boldsymbol{y} = 0.5\boldsymbol{x}^{(1)} + 0.5\boldsymbol{x}^{(2)} + 0.9\boldsymbol{d}^{(1)}.$$



The last geometric definitions we will need to discuss the Simplex method are boundary and interior points.

Definition 5.3 (Def. 5.2 from the textbook)**.** *A feasible solution to a linear program is a boundary point if at least one inequality constraint that can be strict for some feasible solutions is satisfied at equality at a given point. A feasible solution is an interior point if no such inequalities are strict.*

Recall that the boundary of the polyhedron is defined by inequality constraints, and the boundary is determined by these constraints at $=$. Then, for every boundary point there is at least one active constraint.

Let's see an example.

Example 5.1. *Consider the polyhedron defined by:*

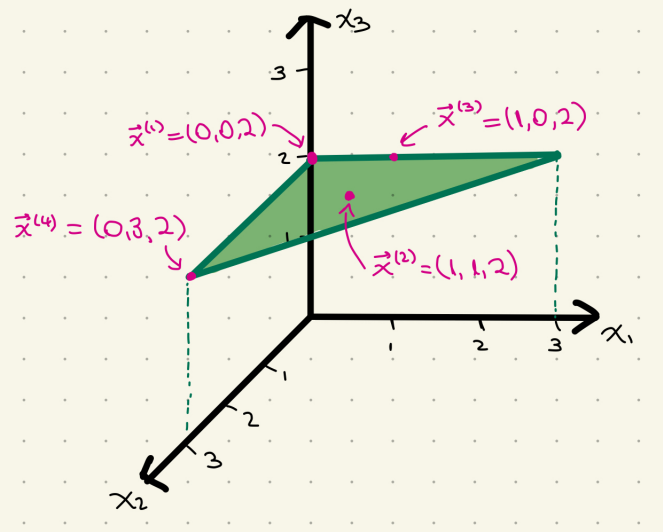
$$F = \left\{\boldsymbol{x} \in \mathbb{R}^3 : \; x_1 + x_2 \leq 3, \, x_1 \geq 0, \, x_2 \geq 0, \, x_3 = 2\right\}$$

and consider the points

$$\boldsymbol{x}^{(1)} = (0, 0, 2), \; \boldsymbol{x}^{(2)} = (1, 1, 2), \; \boldsymbol{x}^{(3)} = (1, 0, 2), \; \boldsymbol{x}^{(4)} = (0, 3, 2).$$

Which of these points are interior, boundary and extreme points? For each point, identify the active constraints.

Solution. We draw the set F and the points to decide whether they are interior, boundary and/or extreme points.



As we see from the figure:

- $\boldsymbol{x}^{(1)}$ is a boundary and an extreme point and the constraints $x_1 \geq 0$ and $x_2 \geq 0$ are active
- $\boldsymbol{x}^{(2)}$ is an interior point and there are no active constraints (among the inequality constraints)
- $\boldsymbol{x}^{(3)}$ is a boundary point and the only active constraint is $x_2 \geq 0$
- $\boldsymbol{x}^{(4)}$ is a boundary and an extreme point, and the active constraints are $x_1 \geq 0$ and $x_1 + x_2 \leq 3$

□

Optimal points in linear programs

Recall the definition of local and global optima from Section 3. Also, remember that global optima are always local, but local optima may not be global. The following principle is one of the reasons why we care so much about linear programs.

Principle 5.1 (Principles 5.3–5.5 from the textbook)**.** *In linear programs*

(i) Every local optimal solution is global.

(ii) Unless the objective function is constant, every optimal solution to an LP will occur at a boundary point of the feasible set.

(iii) If a linear program has a unique optimal solution, that optimum must occur at an extreme point of the feasible region

(iv) If a linear program has at least one optimal solution (maybe multiple), it has at least one at an extreme point of the feasible region.

In simple words, linear programs always have at least one optimal solution at an extreme point (unless they are unbounded).

We won't mathematically prove any of these principles, but observe that these are exactly the patterns we had when we studied graphical solutions.

Example 5.2. *For the points indicated in Example 5.1, indicate if they can be optimal or uniquely optimal in a linear program under appropriate objective functions.*

Solution. We analyze each of the points:

- $\boldsymbol{x}^{(1)}$ is an extreme point and, hence, it can be a unique optimal solution
- $\boldsymbol{x}^{(2)}$ is an interior point, so it cannot be an optimal solution unless the objective function is constant

- $\boldsymbol{x}^{(3)}$ is a boundary point, but not an extreme point. Then, it can be an optimal solution but it will not be a unique optimal solution.
- $\boldsymbol{x}^{(4)}$ is an extreme point, so it can also be a unique optimal solution

□

Thanks to Principle 5.1, in linear programs, it suffices to search for optimal solutions in the extreme points. Once we get an extreme point that is a local optimal solution (i.e., whose value is better than its neighbors), we know we already found the global optima.

Standard form

We already know what is a linear program, and we have established that the objective function and the constraints need to be linear functions. However, we can still write them in several formats. We now introduce the standard form, which is the specific format we will use for Simplex algorithm. As we will see later, we use the standard form to easily compute extreme points of the feasible set.

Definition 5.4 (Def. 5.6 from textbook)**.** *Linear programs in standard form satisfy the following properties:*

(1) The objective function is simplified, i.e., the variables appear at most once

(2) They have only nonnegative variables

(3) The constraints are simplified, all the variables appear on the left-hand side and the constant term (possibly 0) on the right-hand side

(4) They have only equality constraints

Now the main question to answer is if every linear problem can be written in standard form. The answer is yes. However, we might need to add some additional variables (called slack variables) to satisfy the standard form rules.

Example 5.3. *Write the following optimization problem in standard form:*

$$\begin{aligned} \min \quad & 12(x_1 + x_2) - 3(x_1 + x_2 + x_3) \\ s.t. \quad & x_1 + x_2 + x_3 \leq 1000 \\ & x_1 \geq 0.6(x_1 + x_2) \\ & x_1 \geq 0 \\ & x_2 \leq 0 \end{aligned}$$

Solution. We first simplify the objective function as follows:

$$12(x_1 + x_2) - 3(x_1 + x_2 + x_3) = 9x_1 + 9x_2 - 3x_3$$

Now let's go through the constraints.

- The first constraint is "almost" in standard form: it has all the variables on the left-hand side and the constant on the right-hand side. The only part we need to take care of is the inequality. Observe that if $a \leq b$, then there must exist a nonnegative number c such that $a + c = b$. Using this idea, we rewrite the first constraint using the slack variable x_4 as follows:

$$x_1 + x_2 + x_3 \leq 1000 \quad \iff \quad \begin{cases} x_1 + x_2 + x_3 + x_4 = 1000 \\ x_4 \geq 0 \end{cases}$$

- The second constraint needs to be simplified, we need to move all the variables to the left-hand side and we need to take care of the inequality. Let's go step by step. Simplifying and reorganizing all the variables to the left-hand side we obtain:

$$x_1 \geq 0.6(x_1 + x_2) \quad \iff \quad 0.4x_1 - 0.6x_2 \geq 0$$

Now we take care of the inequality. Similarly to the previous constraint, observe that if $a \geq b$, then there must exist a nonnegative number c such that $a - c = b$. Then, we include the slack variable x_5 as follows:

$$0.4x_1 - 0.6x_2 \geq 0 \quad \Longleftrightarrow \quad \begin{cases} 0.4x_1 - 0.6x_2 - x_5 = 0 \\ x_5 \geq 0 \end{cases}$$

Then, when we write the complete standard form of the linear program, we write

$$\begin{aligned} 0.4x_1 - 0.6x_2 - x_5 &= 0 \\ x_5 &\geq 0 \end{aligned}$$

instead of $x_1 \geq 0.6(x_1 + x_2)$

- The next constraint is $x_1 \geq 0$, which is already in standard form.
- The next constraint is $x_2 \leq 0$. Since all the variables need to be nonnegative, we define a new slack variable x_6 that satisfies $x_6 = -x_2$. Then,we replace all the x_2s in the formulation by $-x_6$.
- Finally, the variable x_3 is unrestricted. Then, we include two slack variables, x_7 and x_8 and define $x_3 = x_7 - x_8$, where x_7 and x_8 are nonnegative. Then we replace any appearance of x_3 by $x_7 - x_8$.

Applying all the changes, we obtain the following linear program:

$$\begin{aligned} \min \quad & 9x_1 - 9x_6 - 3x_7 + 3x_8 \\ \text{s.t.} \quad & x_1 - x_6 + x_7 - x_8 + x_4 = 1000 \\ & 0.4x_1 + 0.6x_6 - x_5 = 0 \\ & x_1, x_4, x_5, x_6, x_7, x_8 \geq 0 \end{aligned}$$

□

Since the standard form is so specifically defined, we can actually give a name to each part.

Definition 5.5 (Def. 5.11 from the textbook)**.** *In the usual matrix notation, the linear programming standard form is:*

$$\begin{aligned} \min(or\ \max) \quad & \boldsymbol{c}^T \boldsymbol{x} \\ s.t. \quad & A\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

Example 5.4. *Write the vectors* $\boldsymbol{c}$, $\boldsymbol{b}$ *and the matrix* A *in Example 5.3.*

Solution. The vector $\boldsymbol{c}$ is the vector of weights of the objective function. Then, we obtain

$$\boldsymbol{c} = (9, 0, 0, 0, 0, -9, -3, 3)$$

The vector $\boldsymbol{b}$ represents the right-hand side of the main constraints (i.e., the constraints that we wrote as $=$). We obtain

$$\boldsymbol{b} = (1000, 0)$$

Finally, the matrix A is represents the weights of the variables in the $=$ constraints on the left-hand side. We obtain

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & -1 & 1 & -1 \\ 0.4 & 0 & 0 & 0 & -1 & 0.6 & 0 & 0 \end{bmatrix}$$

□

5.2 Extreme-point search and basic solutions

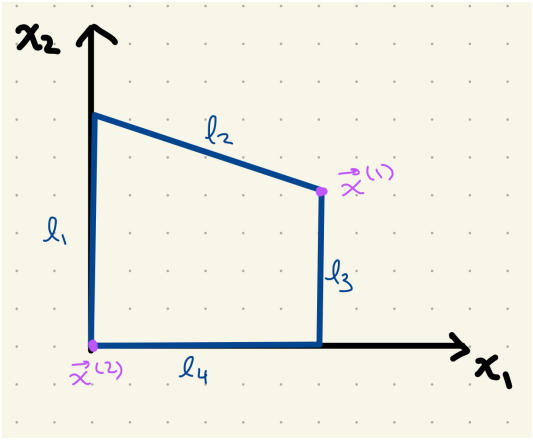
We started this chapter discussing why linear programs are so easy to solve. Summarizing, the main advantage is that we only need to search over the extreme points to find the global optimal solution. Then we learned an alternative way of writing linear programs, where all the main constraints are equalities and all the variables are nonnegative (the standard form). In this subsection we will learn the relationship between the standard form and the extreme points of the original linear program.

As we saw before, extreme points are feasible points that cannot be written as a weighted sum of other points in the feasible set. In practice, when we want to find them in a picture, we look for the vertices. These vertices are characterized by the intersection of 2 inequalities in 2D. But what happens in higher dimensions? How do we find these extreme points?

Principle 5.2 (based on Principle 5.12 from the textbook)**.** *Every extreme point solution $\hat{\boldsymbol{x}}$ to a linear problem with n decision variables (and hence, with a feasible set $F \subset \mathbb{R}^n$) is determined by n constraints that are simultaneously active only at $\hat{\boldsymbol{x}}$.*

Let's see a pictorial example.

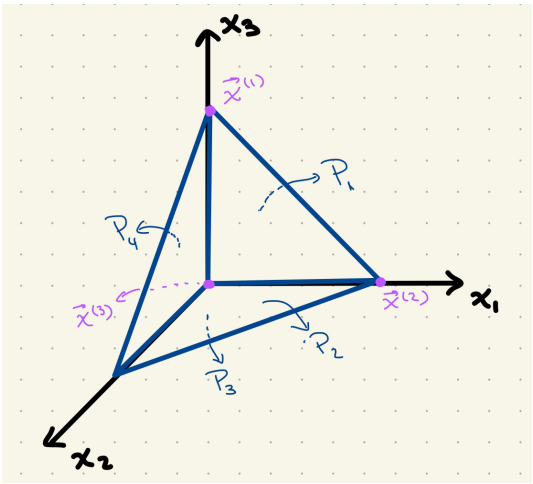
Example 5.5. *Consider the following feasible set in 2D, i.e., for which $n = 2$:*



In the figure, each of the lines ℓ_i for $i \in \{1, 2, 3, 4\}$ represents a constraint and for every point in the line, the constraint defining the line is active. Then, at the extreme point $\boldsymbol{x}^{(1)}$ there are two active constraints: ℓ_2 and ℓ_3. Similarly, at the extreme point $\boldsymbol{x}^{(2)}$ the active constraints are ℓ_1 and ℓ_4.

Now let's go a bit further, and let's explore an example in 3D.

Example 5.6. *Consider the following feasible set, for which $n = 3$.*



In this case, the constraints define planes instead of lines. Then, the extreme point $\boldsymbol{x}^{(1)}$ is at the intersection of the planes P_1, P_2, P_4, the extreme point $\boldsymbol{x}^{(2)}$ is at the intersection of the planes P_1, P_2, P_3, and the extreme point $\boldsymbol{x}^{(3)}$ is at the intersection of the planes P_1, P_3, P_4.

We already know that we will always find an optimal solution at one of the extreme points of the feasible set. The question now is how to search for them. If we start from any extreme point of the feasible region and evaluate the objective function, it would be easy to compare it with the neighboring extreme points. In the following definition we make more concrete the idea of a "neighboring" extreme points.

Definition 5.6 (Def. 5.13 and 5.14 from the textbook)**.** *(i) An edge of the feasible region for a linear program is a 1-dimensional set of feasible points along a line determined by a collection of active constraints.*

(ii) Extreme points of an LP feasible set are adjacent if they are determined by active constraint sets differing only in one element.

For example, in Example 5.5 the extreme points $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ are **not** adjacent and the edges are the lines defined by each constraint, that is, ℓ_1, ℓ_2, ℓ_3 and ℓ_4.

In Example 5.6 the extreme points $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ are adjacent since they share the active constraints P_1, P_2. They differ in that P_4 is active for $\boldsymbol{x}^{(1)}$ and not for $\boldsymbol{x}^{(2)}$, and P_3 is active for $\boldsymbol{x}^{(2)}$ and not $\boldsymbol{x}^{(1)}$. The edges in this example are the intersection of two constraints. For example, there is an edge at the intersection of P_1 and P_2.

From linear algebra, you might remember that the intersection of n equations in $\mathbb{R}^n$ set yields a point and that the intersection of $n-1$ equations in $\mathbb{R}^n$ yields a line. This is exactly the same, but we are assigning the terms "extreme point" and "edge" to these objects instead.

Based on the definition of edge and adjacent extreme point, we can conclude the following key properties.

Principle 5.3.

(i) Adjacent extreme points are separated by an edge

(ii) The edge that separates adjacent extreme points must be one of the active constraints that both extreme points share.

Basic solutions

From now on we will work with problems in the standard form, that is, problems of the form

$$\begin{aligned} \max / \min \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_{i,j} x_j = b_i \qquad \forall i \in \{1, \dots, m\} \\ & x_j \geq 0 \qquad \forall j \in \{1, \dots, n\} \end{aligned}$$

or in matrix form:

$$\begin{aligned} \max / \min \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} \quad & A\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0}, \end{aligned}$$

where $\boldsymbol{x}, \boldsymbol{c}$ are vectors in $\mathbb{R}^n$, A is an $m \times n$ matrix (m rows and n columns) and $\boldsymbol{b}$ is a vector in $\mathbb{R}^m$.

Definition 5.7. *A basic solution to a linear program in standard form is one obtained by fixing enough variables to 0 so that the equality constraints of the model can be solved uniquely for the remaining values.*

The variables set to zero are called nonbasic and the ones obtained by solving the equalities are called basic. Further, the set of basic variables is called basis.

Let's do an example.

Example 5.7. *Consider the following linear program*

$$\begin{aligned} \max \quad & 12x_1 + 9x_2 \\ s.t. \quad & x_1 \leq 1000 \\ & x_2 \leq 1500 \\ & x_1 + x_2 \leq 1750 \\ & 4x_1 + 2x_2 \leq 4800 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Compute a basic solution, determine which variables are basic/nonbasic at that basic solution and identify the corresponding extreme point in a plot.

Solution. We first draw the feasible region. We obtain

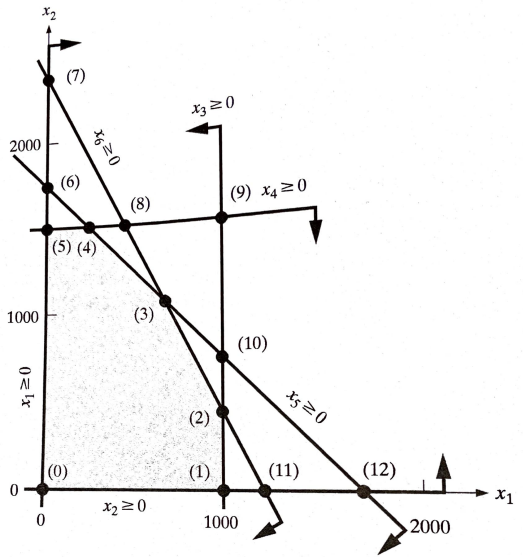


Figure 5.4 from the textbook

Now let's compute the standard form. We obtain

$$\begin{aligned} \max \quad & 12x_1 + 9x_2 \\ \text{s.t.} \quad & x_1 + x_3 = 1000 \\ & x_2 + x_4 = 1500 \\ & x_1 + x_2 + x_5 = 1750 \\ & 4x_1 + 2x_2 + x_6 = 4800 \\ & x_i \geq 0 \qquad \forall i \in \{1, 2, 3, 4, 5, 6\} \end{aligned}$$

Observe that each of the inequality constraints is associated with one of the slack variables we added to compute the standard form and a slack variable being positive means that the corresponding inequality constraint is not active.

Now let's compute a basic solution and detect the basic/nonbasic variables. Since there are 4 equality constraints and 6 variables, we need to set $6-4 = 2$ variables to zero in order to find a unique solution to the system of equations generated by the constraints. Let $x_5 = x_6 = 0$. Then, x_5 and x_6 are the nonbasic variables and x_1, x_2, x_3, x_4 are basic variables.

Setting $x_5 = x_6 = 0$ we obtain the following system of equations:

$$\begin{cases} x_1 + x_3 = 1000 \\ x_2 + x_4 = 1500 \\ x_1 + x_2 = 1750 \\ 4x_1 + 2x_2 = 4800 \end{cases}$$

Solving the system of equations yields

$$\boldsymbol{x} = (650, 1100, 350, 400, 0, 0)$$

which corresponds to the extreme point (3) marked in the plot above. Observe that the constraints linked to the nonbasic variables x_5 and x_6 are active at point (3) and the rest are not active.

From the example above, we observe that we need to solve a system of equations that involves the basic solutions. Hence, if we choose the slack variables as our basic variables, we don't really need to solve. Let's do it. If $x_1 = x_2 = 0$ (i.e. the nonbasic variables), then we obtain the following system of equations:

$$\begin{cases} x_3 = 1000 \\ x_4 = 1500 \\ x_5 = 1750 \\ x_6 = 4800 \end{cases}$$

which is already solved. This basic solution corresponds to the extreme point labeled as (0). □

From the example above it seems like we can choose any set of nonbasic variables and we will obtain a basic solution. However, this is not the case. The choice of the nonbasic variables needs to lead to a system of equations with a unique solution. In the following principle we state this fact more formally.

Principle 5.4 (Principle 5.16 from the textbook)**.** *A basic solution exists if and only if the columns of the matrix A corresponding to the basic variables form a basis of $\mathbb{R}^m$, where m is the number of main constraints of the linear program.*

In the example above, if we choose x_2, x_4 as nonbasic variables we obtain the following system of equations:

$$\begin{cases} x_1 + x_3 = 1000 \\ 0 = 1500 \\ x_1 + x_5 = 1750 \\ 4x_1 + x_6 = 4800 \end{cases}$$

and the second equation does not have a solution. If we look at the plot of the feasible region, we observe that if $x_4 = 0$, then $x_2 = 1500$. Hence, $x_2 \geq 0$ cannot be active.

The existence of a basic solution given our choice of nonbasic variables is yet another reason why we would like to choose the slack variables as basic solutions. Since every constraint has a different slack variable, the columns of A associated to them are always linearly independent (recall what we've learned in linear algebra).

Basic feasible solutions

So far we learned how to compute basic solutions, and we observed that they correspond to points. However, there is nothing in the definition of a basic solution that ensures that the point we find is feasible.

For example, if we let x_3, x_4 be nonbasic variables (i.e., we set $x_3 = x_4 = 0$) we obtain the following system of equations

$$\begin{cases} x_1 = 1000 \\ x_2 = 1500 \\ x_1 + x_2 + x_5 = 1750 \\ 4x_1 + 2x_2 + x_6 = 4800 \end{cases}$$

that yields

$$\boldsymbol{x} = (1000, 1500, 0, 0, -750, -2200)$$

In other words, the constraints $x_5 \geq 0$ and $x_6 \geq 0$ are violated and, hence, $\boldsymbol{x}$ is not feasible. This is consistent with the fact that intersecting the lines corresponding to the first and second constraint yields point (9) in the plot, which is outside the feasible region.

Definition 5.8 (Principle 5.17 from the textbook)**.** *A basic feasible solution to a linear program in standard form is a basic solution that satisfies the nonnegativity constraints.*

Then, what is the relationship between extreme points and basic solutions?

Principle 5.5 (Principle 5.18 from the textbook)**.** *The basic feasible solutions of a linear program in standard form are exactly the extreme points of its feasible region.*

Before moving on to the Simplex algorithm, let's do an example to summarize what we've learned so far.

Example 5.8 (Example 5.10 from the textbook)**.** *Consider a linear program with the following set of constraints in standard form:*

$$\begin{aligned} -x_1 + x_2 - x_3 &= 0 \\ x_1 + x_4 &= 2 \\ x_2 + x_5 &= 3 \\ x_i \geq 0 \qquad & \forall i \in \{1,2,3,4,5\} \end{aligned}$$

Compute the basic solution corresponding to the bases $B_1 = \{x_3, x_4, x_5\}$ *and* $B_2 = \{x_1, x_2, x_4\}$*, and determine if they are basic feasible solutions.*

Solution. We start with basis $B_1 = \{x_3, x_4, x_5\}$, that is, we set $x_1 = x_2 = 0$ and obtain the following system of equations:

$$\begin{cases} -x_3 = 0 \\ x_4 = 2 \\ x_5 = 3 \end{cases}$$

Then,

$$\boldsymbol{x}^{B_1} = (0, 0, 0, 2, 3)$$

and since all its elements are nonnegative, we conclude that $\boldsymbol{x}^{B_1}$ is a basic feasible solution.

Now we compute the basic solution associated to the basis $B_2 = \{x_1, x_2, x_4\}$, that is, we set $x_3 = x_5 = 0$. We obtain the following system of equations:

$$\begin{cases} -x_1 + x_2 = 0 \\ x_1 + x_4 = 2 \\ x_2 = 3 \end{cases}$$

Then, the basic solution associated to the basis B_2 is

$$\boldsymbol{x}^{B_2} = (2, 2, 0, -1, 0)$$

Since the fourth element is negative, $\boldsymbol{x}^{B_2}$ is not a basic feasible solution. □

5.3 The Simplex Algorithm

Simplex is an algorithm to compute optimal solutions in linear programs, and it exploits the properties of linear programs to search for the optimal solution efficiently. As any "improving search" algorithm, we move from solution to solution retaining feasibility and improving the objective function value until we find an optimal solution.

To completely define the algorithm, we answer the following questions in the rest of this section:

1. Where do we start the search?
2. How do we move in an improving and feasible direction?
3. When do we stop searching?

Since simplex is specific to linear programs and we know that linear programs always have an optimal solution at an extreme point, we move from extreme point to extreme point. More specifically, we move among adjacent basic feasible solutions until we find an optimal solution. We give more details in the subsequent subsections.

As usual, we first establish the notation. The standard display of a linear program uses the vectors $\boldsymbol{c}$, $\boldsymbol{b}$ and the matrix A of the standard form of the linear problem. We construct a table that shows all the BFS we visit, and the table grows as we iterate to find an optimal solution.

Let's continue our working example (Example 5.7), which had standard form:

$$\begin{aligned} \max \quad & 12x_1 + 9x_2 \\ \text{s.t.} \quad & x_1 + x_3 = 1000 \\ & x_2 + x_4 = 1500 \\ & x_1 + x_2 + x_5 = 1750 \\ & 4x_1 + 2x_2 + x_6 = 4800 \\ & x_i \geq 0 \qquad \forall i \in \{1, 2, 3, 4, 5, 6\} \end{aligned}$$

We display it in the following table:

	x_1	x_2	x_3	x_4	x_5	x_6	
$\max \boldsymbol{c}$	12	9	0	0	0	0	$\boldsymbol{b}$
A	1	0	1	0	0	0	1000
	0	1	0	1	0	0	1500
	1	1	0	0	1	0	1750
	4	2	0	0	0	1	4800

Question 1: Initial basic feasible solution

Our first task is to find an extreme point to start the search. To compute the initial extreme point, we apply what we learned about BFS. Any BFS is okay.

We already computed some BFS to this problem, so we just choose one. We add the information about the BFS to the table:

	x_1	x_2	x_3	x_4	x_5	x_6	
$\max \boldsymbol{c}$	12	9	0	0	0	0	$\boldsymbol{b}$
A	1	0	1	0	0	0	1000
	0	1	0	1	0	0	1500
	1	1	0	0	1	0	1750
	4	2	0	0	0	1	4800
	N	N	B	B	B	B	
$\boldsymbol{x}^{(0)}$	0	0	1000	1500	1750	4800	

Question 2: Simplex improving directions and step size

The next step is to move to a better basic feasible solution. We want to move to an adjacent BFS, and recall that adjacent extreme points (or BFS) share all but one active constraint. Since we have the problem in standard form, the only constraints that can be active are the nonnegativity constraints and the variables with active constraints are the nonbasic variables. Then, we look at the current nonbasic variables and we evaluate possible new nonbasic variables, one at a time.

Principle 5.6 (Principle 5.20 from the textbook)**.** *The Simplex directions are constructed by:*

(i) Increasing the value of a single nonbasic variable

(ii) Leaving all the other nonbasic variables unchanged

(iii) Computing the corresponding changes in the basic variables to preserve the equality constraints

Observe that there is one simplex direction for each nonbasic variable.

Let's use $\Delta \boldsymbol{x}$ to denote the simplex direction. To ensure feasibility, we need to make sure that the new point also satisfies the equality constraints $A\boldsymbol{x} = \boldsymbol{b}$. Let $\boldsymbol{x}^{(1)}$ be the new point we will evaluate. Then,

$$\boldsymbol{x}^{(1)} \leftarrow \boldsymbol{x}^{(0)} + \Delta \boldsymbol{x}$$

and we need

$$A\boldsymbol{x}^{(1)} = \boldsymbol{b} \qquad \iff \qquad A(\boldsymbol{x}^{(0)} + \Delta \boldsymbol{x}) = \boldsymbol{b} \qquad \iff \qquad A\Delta \boldsymbol{x} = \boldsymbol{0}$$

Hence, the simplex direction $\Delta \boldsymbol{x}$ must satisfy $A\Delta \boldsymbol{x} = \mathbf{0}$ to be feasible.

In our example we have two nonbasic variables. If we increase the value of x_1, then $\Delta x_1 = 1$ and $\Delta x_2 = 0$. Then, we solve the following system of equations to compute the rest of the elements of $\Delta \boldsymbol{x}$:

$$
\begin{aligned}
A\Delta \boldsymbol{x} = \mathbf{0} \quad &\iff \quad \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 4 & 2 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 1 \\ 0 \\ \Delta x_3 \\ \Delta x_4 \\ \Delta x_5 \\ \Delta x_6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \\
&\iff \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \Delta x_3 \\ \Delta x_4 \\ \Delta x_5 \\ \Delta x_6 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ -1 \\ -4 \end{pmatrix} \\
&\iff \quad \Delta \boldsymbol{x} = \begin{pmatrix} -1 \\ 0 \\ -1 \\ -4 \end{pmatrix}
\end{aligned}
$$

Similarly, if we increase the value of the nonbasic variable x_2 and we keep $x_1 = 0$, we obtain the system of equations

$$
\begin{aligned}
A\Delta \boldsymbol{x} = \mathbf{0} \quad &\iff \quad \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 4 & 2 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 0 \\ 1 \\ \Delta x_3 \\ \Delta x_4 \\ \Delta x_5 \\ \Delta x_6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \\
&\iff \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \Delta x_3 \\ \Delta x_4 \\ \Delta x_5 \\ \Delta x_6 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \\ -1 \\ -2 \end{pmatrix} \\
&\iff \quad \Delta \boldsymbol{x} = \begin{pmatrix} 0 \\ -1 \\ -1 \\ -2 \end{pmatrix}
\end{aligned}
$$

We add this information to the table, as follows:

	x_1	x_2	x_3	x_4	x_5	x_6	
$\max \boldsymbol{c}$	12	9	0	0	0	0	$\boldsymbol{b}$
A	1	0	1	0	0	0	1000
	0	1	0	1	0	0	1500
	1	1	0	0	1	0	1750
	4	2	0	0	0	1	4800
	N	N	B	B	B	B	
$\boldsymbol{x}^{(0)}$	0	0	1000	1500	1750	4800	
$\Delta \boldsymbol{x}$ for x_1	1	0	-1	0	-1	-4	
$\Delta \boldsymbol{x}$ for x_2	0	1	0	-1	-1	-2	

Will we always obtain a unique direction? Yes, because by increasing the value of one of the nonbasic variables to 0 and leaving all the other nonbasic variables in 0 we are creating an $n \times n$ system of equations with linearly independent columns.

Now we know how to compute feasible directions, but how do we know that the objective function value will improve?

Principle 5.7 (Principles 3.20 and 3.21 from the textbook)**.** *A direction $\Delta \boldsymbol{x}$ is improving for a maximizing/minimizing linear program with cost vector $\boldsymbol{c}$ if $\boldsymbol{c}^T \Delta \boldsymbol{x} > 0$/$\boldsymbol{c}^T \Delta \boldsymbol{x} < 0$.*

Since we evaluate multiple directions, we introduce a new concept.

Definition 5.9 (Principle 5.21 from the textbook)**.** *The reduced cost $\bar{c}_j$ associated with increasing the nonbasic variable x_j, also known as reduced cost of x_j, is defined as*

$$\bar{c}_j \triangleq \boldsymbol{c}^T \Delta \boldsymbol{x},$$

where $\Delta \boldsymbol{x}$ is the simplex direction increasing x_j.

Hence, we obtain the following principle for linear programming.

Principle 5.8. *The simplex direction increasing the nonbasic variable x_j is improving for a maximize/minimize linear program if the corresponding reduced cost satisfies $\bar{c}_j > 0$/$\bar{c}_j < 0$.*

In our example, we obtain

$$\bar{c}_1 = 12 > 0 \qquad \text{and} \qquad \bar{c}_2 = 9 > 0.$$

Therefore, both x_1 and x_2 result in an improving direction. We add this information to the last column of our table, as follows:

	x_1	x_2	x_3	x_4	x_5	x_6	
$\max \boldsymbol{c}$	12	9	0	0	0	0	$\boldsymbol{b}$
A	1	0	1	0	0	0	1000
	0	1	0	1	0	0	1500
	1	1	0	0	1	0	1750
	4	2	0	0	0	1	4800
	N	N	B	B	B	B	
$\boldsymbol{x}^{(0)}$	0	0	1000	1500	1750	4800	
$\Delta \boldsymbol{x}$ for x_1	1	0	-1	0	-1	-4	$\bar{c}_1 = 12$
$\Delta \boldsymbol{x}$ for x_2	0	1	0	-1	-1	-2	$\bar{c}_2 = 9$

Any improving direction can be picked, but the next question is how much we can move in the desired improving direction. We would like to take the biggest step that preserves feasibility and improves the objective function value, and also land at another extreme point. Since the gradient of linear programs is constant, an improving direction is always improving. Then, we only need to worry about feasibility. By construction of the simplex direction $\Delta \boldsymbol{x}$, the equality constraints $A\boldsymbol{x} = \boldsymbol{b}$ are always satisfied. Hence, we only need to check feasibility with respect to the nonnegativity constraints. That is, to compute the step size, we find the first basic variable that becomes 0 when moving in $\Delta \boldsymbol{x}$ direction. More formally, we use the minimum ratio criteria described below.

Principle 5.9 (Principles 5.23 and 5.24 of the textbook)**.**

1. *If any component is negative in the improving simplex direction $\Delta \boldsymbol{x}$ at the current basic feasible solution $\boldsymbol{x}^{(t)}$, then simplex search uses the **maximum feasible step of minimum ratio** computation to determine the stepsize λ:*

$$\lambda \triangleq \min \left\{ \frac{x_j}{-\Delta x_j} \,:\, \Delta x_j < 0 \right\}$$

2. *If no component is negative in the improving simplex direction $\Delta \boldsymbol{x}$ at the current basic feasible solution $\boldsymbol{x}^{(t)}$, then the solution can be improved forever in direction $\Delta \boldsymbol{x}$. That is, the linear program is unbounded.*

In our example, let's suppose we choose to increase the value of the nonbasic variable x_1. Then, the step size is

$$\lambda = \min \left\{ \frac{-1000}{-(-1)}, \frac{1750}{-(-1)}, \frac{4800}{-(-4)} \right\} = 1000$$

Then, the new solution is

$$\begin{aligned} \boldsymbol{x}^{(1)} &\leftarrow \boldsymbol{x}^{(0)} + \lambda \Delta \boldsymbol{x} \\ &= (0, 0, 1000, 1500, 1750, 4800) + 1000(1, 0, -1, 0, -1, -4) \\ &= (1000, 0, 0, 500, 750, 800) \end{aligned}$$

Observe that the active nonnegative constraints at $\boldsymbol{x}^{(1)}$ are $x_2 \geq 0$ and $x_3 \geq 0$. Then, the nonbasic variables at the BFS $\boldsymbol{x}^{(1)}$ are x_2 and x_3 and the basis becomes x_1, x_4, x_5, x_6. In other words, we update the basis. More formally, we have the following principle.

Principle 5.10 (Principle 5.25 from the textbook)**.** *After each move of simplex search, the nonbasic variable generating the chosen simplex direction enters the basis, and any of the (possible several) basic variables fixing the step size λ leaves the basis.*

In our example, x_3 leaves the basis and x_1 enters the basis.

In some cases, we obtain $\lambda = 0$ from the minimum ratio computation. In such case, we say that the current basic feasible solution is degenerate. For the time being, we will just classify the case as degenerate and we will study what it means later in the semester.

Now that we formalized the idea of moving from one extreme point to another while making sure that the objective function value improves, we know how to search in an improving direction. The last question to answer is when do we stop searching.

Question 3: Detecting optimal solutions

Since simplex is an improving search algorithm, we know that we are at an optimal solution if none of the possible simplex directions are improving. Formally, we have the following principle.

Principle 5.11. *In a maximization/minimization problem, when simplex reaches a basic feasible solution such that all the nonbasic variables have a negative/positive reduced cost, we know that the current basic feasible solution is an optimal solution.*

Rudimentary simplex algorithm

Let's summarize the step of simplex.

Algorithm 5.1 (Rudimentary simplex search for linear programs)**.**

0. ***Initialization***

   0.1 *Choose any starting feasible basis*

   0.2 *Compute the corresponding basic feasible solution* $\boldsymbol{x}^{(0)}$

   0.3 *Set the solution index* $t \leftarrow 0$.

1. ***Simplex directions***
   For each nonbasic variable x_j*:*

   1.1 *Construct the simplex direction* $\Delta\boldsymbol{x}$ *associated with increasing the nonbasic variable* x_j

   1.2 *Compute the corresponding reduced cost* $\bar{c}_j = \boldsymbol{c}^T \Delta\boldsymbol{x}$.

2. ***Optimality***

   - *If no direction is improving, then stop. Current solution* $\boldsymbol{x}^{(t)}$ *is an optimal solution.*
     In a maximization/minimization problem, no direction is improving when none of the nonbasic variables x_j *satisfies* $\bar{c}_j > 0$/$\bar{c}_j < 0$.
   - *Otherwise, choose any improving simplex direction as* $\Delta\boldsymbol{x}^{(t+1)}$ *and denote the entering basic variable* x_p.

3. ***Step size***

   - *If* $\Delta\boldsymbol{x}^{(t+1)} > \mathbf{0}$*, that is, if all the elements of the improving direction are strictly positive, then stop. The given model is unbounded.*
   - *Otherwise, compute the step size by*

$$\lambda_{t+1} \leftarrow \min\left\{ \frac{x_j^{(t)}}{-\Delta x_j^{(t+1)}} : \Delta x_j^{(t+1)} < 0 \right\}$$

   The minimizer variable x_r *leaves the basis.*

*4. **New point and basis***

4.1 Compute the new solution

$$\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} + \lambda_{t+1}\Delta\boldsymbol{x}^{(t+1)}$$

4.2 Replace x_r by x_p in the basis

4.3 Advance $t \leftarrow t+1$

4.4 Return to step 1.

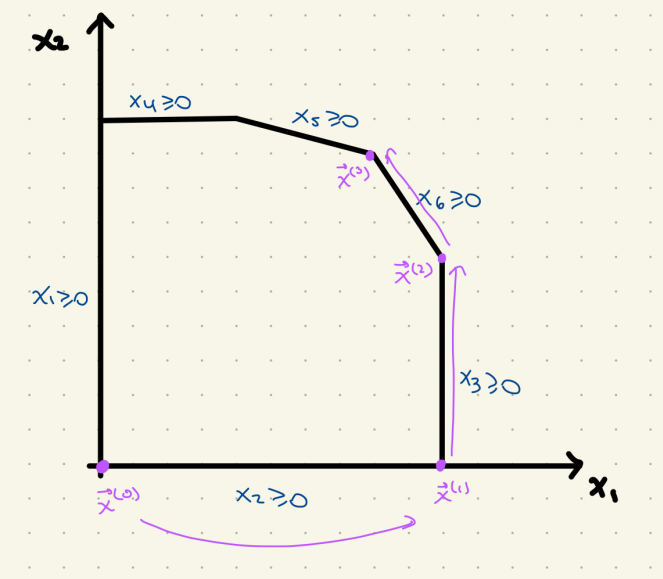
Let's apply the algorithm to our working example. We obtain:

	x_1	x_2	x_3	x_4	x_5	x_6	
$\max \boldsymbol{c}^T\boldsymbol{x}$	12	9	0	0	0	0	$\boldsymbol{b}$
A	1	0	1	0	0	0	1000
	0	1	0	1	0	0	1500
	1	1	0	0	1	0	1750
	4	2	0	0	0	1	4800
$t=0$	N	N	B	B	B	B	
$\boldsymbol{x}^{(0)}$	0	0	1000	1500	1750	4800	$\boldsymbol{c}^T\boldsymbol{x}^{(0)} = 0$
$\Delta\boldsymbol{x}$ for x_1	1	0	-1	0	-1	-4	$\boxed{\bar{c}_1 = 12}$
$\Delta\boldsymbol{x}$ for x_2	0	1	0	-1	-1	-2	$\bar{c}_2 = 9$
Step size	–	–	$\boxed{1000/\text{-}(\text{-}1)}$	–	1750/-(-1)	4800/-(-4)	$\lambda = 1000$
$t=1$	B	N	N	B	B	B	
$\boldsymbol{x}^{(1)}$	1000	0	0	1500	750	800	$\boldsymbol{c}^T\boldsymbol{x}^{(1)} = 12000$
$\Delta\boldsymbol{x}$ for x_2	0	1	0	-1	-1	-2	$\boxed{\bar{c}_2 = 9}$
$\Delta\boldsymbol{x}$ for x_3	-1	0	1	0	1	4	$\bar{c}_3 = -12$
Step size	–	–	–	500/-(-1)	750/-(-1)	$\boxed{800/\text{-}(\text{-}2)}$	$\lambda = 400$
$t=2$	B	B	N	B	B	N	
$\boldsymbol{x}^{(2)}$	1000	400	0	1100	350	0	$\boldsymbol{c}^T\boldsymbol{x}^{(2)} = 15600$
$\Delta\boldsymbol{x}$ for x_3	-1	2	1	-2	-1	0	$\boxed{\bar{c}_3 = 6}$
$\Delta\boldsymbol{x}$ for x_6	0	-1/2	0	1/2	1/2	1	$\bar{c}_6 = -9/2$
Step size	1000/-(-1)	–	–	1100/-(-2)	$\boxed{350/\text{-}(\text{-}1)}$	–	$\lambda = 350$
$t=3$	B	B	B	B	N	N	
$\boldsymbol{x}^{(3)}$	650	1100	350	400	0	0	$\boldsymbol{c}^T\boldsymbol{x}^{(3)} = 17700$
$\Delta\boldsymbol{x}$ for x_5	1	-2	-1	2	1	0	$\bar{c}_5 = -6$
$\Delta\boldsymbol{x}$ for x_6	-1/2	1/2	1/2	-1/2	0	1	$\bar{c}_6 = -3/2$

Since the reduced costs are negative in the last basic feasible solution, and we are maximizing, we found the optimal solution. The optimal solution is

$$\boldsymbol{x}^* = (650, 1100, 350, 400, 0, 0)$$

and the optimal value is 17700. Graphically, we went through the following extreme points:



Before moving on to improvements to the rudimentary simplex algorithm presented above, we solve one more example.

Example 5.9. *Consider the following optimization problem.*

$$\begin{aligned} \max \quad & x_1 + x_2 \\ s.t. \quad & -2x_1 + x_2 \leq 1 \\ & x_1 - \alpha x_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

where $\alpha \geq 0$ is a parameter. Use simplex algorithm to determine for which values of α the problem has a unique optimal solution, and for which values is unbounded.

Solution. We first write the problem in standard form:

$$\begin{aligned} \max \quad & x_1 + x_2 \\ \text{s.t.} \quad & -2x_1 + x_2 + x_3 = 1 \\ & x_1 - \alpha x_2 + x_4 = 1 \\ & x_i \geq 0 \qquad \forall i \in \{1, 2, 3, 4\} \end{aligned}$$

Now we can run the rudimentary simplex algorithm. It is easy to see that an initial bfs is $\boldsymbol{x}^{(0)} = (0, 0, 1, 1)$, where x_1, x_2 are the nonbasic variables and x_3, x_4 are the basic variables. We can no start building the simplex table, as follows:

	x_1	x_2	x_3	x_4	
$\max \boldsymbol{c}^T \boldsymbol{x}$	1	1	0	0	$\boldsymbol{b}$
A	-2	1	1	0	1
	1	$-\alpha$	0	1	1
$t = 0$	N	N	B	B	
$\boldsymbol{x}^{(0)}$	0	0	1	1	$\boldsymbol{c}^T \boldsymbol{x}^{(0)} = 0$

Now we compute the simplex directions associated to each of the nonbasic variables, and their respective reduced costs. We obtain

	x_1	x_2	x_3	x_4	
$\max \boldsymbol{c}^T\boldsymbol{x}$	1	1	0	0	$\boldsymbol{b}$
A	-2	1	1	0	1
	1	$-\alpha$	0	1	1
$t=0$	N	N	B	B	
$\boldsymbol{x}^{(0)}$	0	0	1	1	$\boldsymbol{c}^T\boldsymbol{x}^{(0)}=0$
$\Delta\boldsymbol{x}$, x_1	1	0	2	-1	$\bar{c}_1=1$
$\Delta\boldsymbol{x}$, x_2	0	1	-1	α	$\bar{c}_2=1$

Since both of the reduced costs are positive, and we are maximizing, both of the simplex directions we computed are improving. We may choose any to continue. In this case, we choose x_1. Then, we compute the step size using the min-ratio criteria. We obtain

	x_1	x_2	x_3	x_4	
$\max \boldsymbol{c}^T\boldsymbol{x}$	1	1	0	0	$\boldsymbol{b}$
A	-2	1	1	0	1
	1	$-\alpha$	0	1	1
$t=0$	N	N	B	B	
$\boldsymbol{x}^{(0)}$	0	0	1	1	$\boldsymbol{c}^T\boldsymbol{x}^{(0)}=0$
$\Delta\boldsymbol{x}$, x_1	1	0	2	-1	$\boxed{\bar{c}_1=1}$
$\Delta\boldsymbol{x}$, x_2	0	1	-1	α	$\bar{c}_2=1$
λ	–	–	–	$\boxed{1}$	$\lambda=1$

Next, we compute $\boldsymbol{x}^{(1)}=\boldsymbol{x}^{(0)}+\lambda\Delta\boldsymbol{x}^{(1)}$ and the new simplex directions to evaluate if they are improving. We obtain the following updated table:

	x_1	x_2	x_3	x_4	
$\max \boldsymbol{c}^T\boldsymbol{x}$	1	1	0	0	$\boldsymbol{b}$
A	-2	1	1	0	1
	1	$-\alpha$	0	1	1
$t=0$	N	N	B	B	
$\boldsymbol{x}^{(0)}$	0	0	1	1	$\boldsymbol{c}^T\boldsymbol{x}^{(0)}=0$
$\Delta\boldsymbol{x}$, x_1	1	0	2	-1	$\boxed{\bar{c}_1=1}$
$\Delta\boldsymbol{x}$, x_2	0	1	-1	α	$\bar{c}_2=1$
λ	–	–	–	$\boxed{1}$	$\lambda=1$
$t=1$	B	N	B	N	
$\boldsymbol{x}^{(1)}$	1	0	3	0	$\boldsymbol{c}^T\boldsymbol{x}=1$
$\Delta\boldsymbol{x}$, x_2	α	1	$2\alpha-1$	0	$\boxed{\bar{c}_2=\alpha+1}$
$\Delta\boldsymbol{x}$, x_4	-1	0	2	1	$\bar{c}_4=-1$

Since $\alpha \geq 0$, then $\alpha+1>0$. Therefore, the simplex direction associated to the nonbasic variable x_2 is the only improving direction. Hence, we choose

$$\Delta\boldsymbol{x}^{(2)}=(\alpha,1,2\alpha-1,0)$$

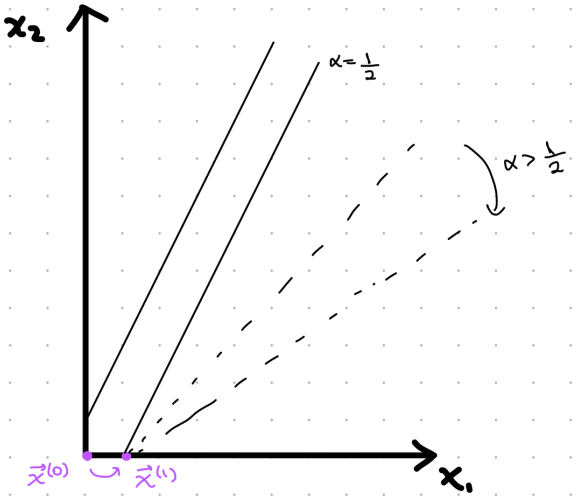
Observe that $\Delta x_1, \Delta x_2, \Delta x_4 \geq 0$. Then, the problem is unbounded if $\Delta x_3 \geq 0$ and has a unique optimal solution if $\Delta x_3 < 0$. That is, if $\alpha \geq 1/2$ the problem is unbounded and if $\alpha < 1/2$ the problem has a unique optimal solution.

To solve for the second case, suppose $\alpha = 1/3$. Then, we obtain the following table

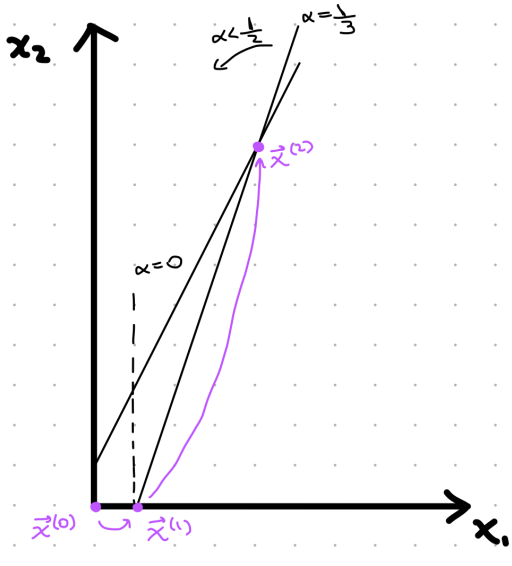
	x_1	x_2	x_3	x_4	
$\max \boldsymbol{c}^T\boldsymbol{x}$	1	1	0	0	$\boldsymbol{b}$
A	-2	1	1	0	1
	1	$-\alpha$	0	1	1
$t=0$	N	N	B	B	
$\boldsymbol{x}^{(0)}$	0	0	1	1	$\boldsymbol{c}^T\boldsymbol{x}^{(0)} = 0$
$\Delta\boldsymbol{x}$, x_1	1	0	2	-1	$\boxed{\bar{c}_1 = 1}$
$\Delta\boldsymbol{x}$, x_2	0	1	-1	α	$\bar{c}_2 = 1$
λ	–	–	–	$\boxed{1}$	$\lambda = 1$
$t=1$	B	N	B	N	
$\boldsymbol{x}^{(1)}$	1	0	3	0	$\boldsymbol{c}^T\boldsymbol{x}^{(1)} = 1$
$\Delta\boldsymbol{x}$, x_2	1/3	1	$-2/3$	0	$\boxed{\bar{c}_2 = 4/3}$
$\Delta\boldsymbol{x}$, x_4	-1	0	2	1	$\bar{c}_4 = -1$
λ	–	–	$\boxed{9}$	–	$\lambda = 9$
$t=2$	B	B	N	N	
$\boldsymbol{x}^{(2)}$	4	9	0	0	$\boldsymbol{c}^T\boldsymbol{x}^{(2)} = 13$
$\Delta\boldsymbol{x}$, x_3	-1	-3	1	0	$\bar{c}_3 = -4$
$\Delta\boldsymbol{x}$, x_4	-3	-6	0	1	$\bar{c}_4 = -9$

Since both reduced costs are negative, we confirm we have found the optimal solution $\boldsymbol{x}^* = (4, 9, 0, 0)$ with optimal value 13.

To close this problem, we show both cases graphically in the following pictures.



(a) $\alpha \geq 1/2$ implies unbounded



(b) $\alpha < 1/2$ implies unique optimal solution

On the left panel we observe that if $\alpha = 1/2$, both inequalities yield parallel lines and, hence, the problem becomes unbounded. For $\alpha > 1/2$ the second line become more and more 'horizontal', so the problem keeps being unbounded. Observe that when $\alpha = 1/2$ the constraints do not intersect and when $\alpha > 1/2$, they intersect at negative values of x_1, x_2.

On the right panel, instead, we plot 2 cases of $\alpha > 1/2$, which implies that the line is more 'vertical'. Under this condition, the constraints intersect for positive x_1, x_2 and, therefore, it bounds the optimal solution. □

5.4 Two phase simplex

Since simplex is about searching for an optimal solution efficiently, it is not always important where to start the search. However, it is important to compute a starting point quickly. An easy choice of the basis are the slack variables that we included in the model to compute the standard form. However, this algorithm may fail in two cases:

- If we don't have slack variables for all the constraints.
  We can always choose variables that appear in only one constraint, but, again, this may not be possible.
- We may obtain a basic solution that is not feasible.
  For example, if we have a linear program that only has $\geq$ constraints and the right hand side vector is positive, using the slack variables as basis will give us negative values. Hence, we obtain a basic solution that is not a BFS.

In this subsection we study a method that will never fail and, further, it will help us detect when our linear program is infeasible. The main idea is to add an artificial variable to every constraint of the standard form and solve an auxiliary linear program to minimize their value. If the problem is infeasible, this minimization will yield a positive optimal objective function value. If the problem is feasible, we will obtain an initial BFS. This algorithm to find an initial BFS is called phase I.

Formally, we have the following principle.

Principle 5.12 (Principle 5.33 from the textbook)**.** *A starting basis for simplex can be obtained by making basic one variable per row, with that variable satisfying:*

(i) It only has a nonzero coefficient in one row

(ii) The coefficient sign matches the right-hand side sign

If no standard-form variable meets these requirements, an artificial variable is introduced.

Let's see some examples.

Example 5.10 (Example 5.17 from the textbook)**.** *Construct a Phase I artificial model to find a BFS for the following linear program in standard form:*

$$\begin{array}{lrrrrrr} \min & 14x_1 & & -9x_3 & +x_4 & & \\ s.t. & 8x_1 & +x_2 & -x_3 & & = & 74 \\ & & 4x_2 & -7x_3 & +x_4 & = & -22 \\ & & x_2 & +x_3 & & = & 11 \\ & x_i \geq 0 & \forall i & & & & \end{array}$$

Solution. We choose one variable per row using the principle.

- For the first constraint (row), we choose x_1.
- For the second row, the only variable that has a nonzero coefficient only once is x_4. However, x_4 has a $+1$ coefficient and the right-hand side of the second constraint is -22. Hence, we introduce the artificial variable x_5 and subtract on the left-hand side of the second constraint.
- For the third row, all the original variables appear in all the constraints. Since the right-hand side is positive, we introduce the artificial variable x_6 and we add it to the left-hand side.

We obtain the following phase-I model:

$$\begin{array}{lrrrrrrr} \min & & & & & x_5 & +x_6 & \\ \text{s.t.} & 8x_1 & +x_2 & -x_3 & & & = & 74 \\ & & 4x_2 & -7x_3 & +x_4 & -x_5 & = & -22 \\ & & x_2 & +x_3 & & & +x_6 & = & 11 \\ & x_i \geq 0 & \forall i & & & & & \end{array}$$

For this problem, we choose the nonbasic variable x_2, x_3, x_4 and the basic variable x_1, x_5, x_6. Hence, the initial solution for the artificial model is

$$\boldsymbol{x}^a = \left(\frac{74}{8}, 0, 0, 0, 22, 11\right)$$

Then, we can run simplex and obtain that the optimal value is 0 and the optimal solution is

$$(\boldsymbol{x}^a)^* = (9.375,\ 5,\ 6,\ 0,\ 0,\ 0)$$

Hence, an initial solution for the original problem is

$$\boldsymbol{x} = (9.375,\ 5,\ 6,\ 0)$$

that is, we simply eliminate the artificial variables. □

Example 5.11 (Example 5.18 from the textbook)**.** *Use Phase I simplex to determine if the following LP is infeasible.*

$$\begin{aligned} \max \quad & 8x_1 + 11x_2 \\ s.t. \quad & x_1 + x_2 \leq 2 \\ & x_1 + x_2 \geq 3 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Solution. We first compute the standard form. We obtain

$$\begin{array}{lrrrrrl} \max & 8x_1 & +11x_2 & & & & \\ \text{s.t.} & x_1 & +x_2 & +x_3 & & = & 2 \\ & x_1 & +x_2 & & -x_4 & = & 3 \\ & x_i \geq 0 & \forall i & & & & \end{array}$$

We only need to include an artificial variable to the second constraint, since the sign of the slack variable is opposite to the sign of the right-hand side and the remaining variables are repeated in the first constraint. Hence, we obtain the following phase-I model:

$$\begin{array}{lrrrrrrl} \min & & & & & x_5 & & \\ \text{s.t.} & x_1 & +x_2 & +x_3 & & & = & 2 \\ & x_1 & +x_2 & & -x_4 & +x_5 & = & 3 \\ & x_i \geq 0 & \forall i & & & & & \end{array}$$

Solving the optimization models yields optimal value 1, and optimal solution

$$(\boldsymbol{x}^a)^* = (0, 2, 0, 0, 1)$$

Since the optimal value of the phase-I model is positive (nonzero), we conclude that the original problem is infeasible.

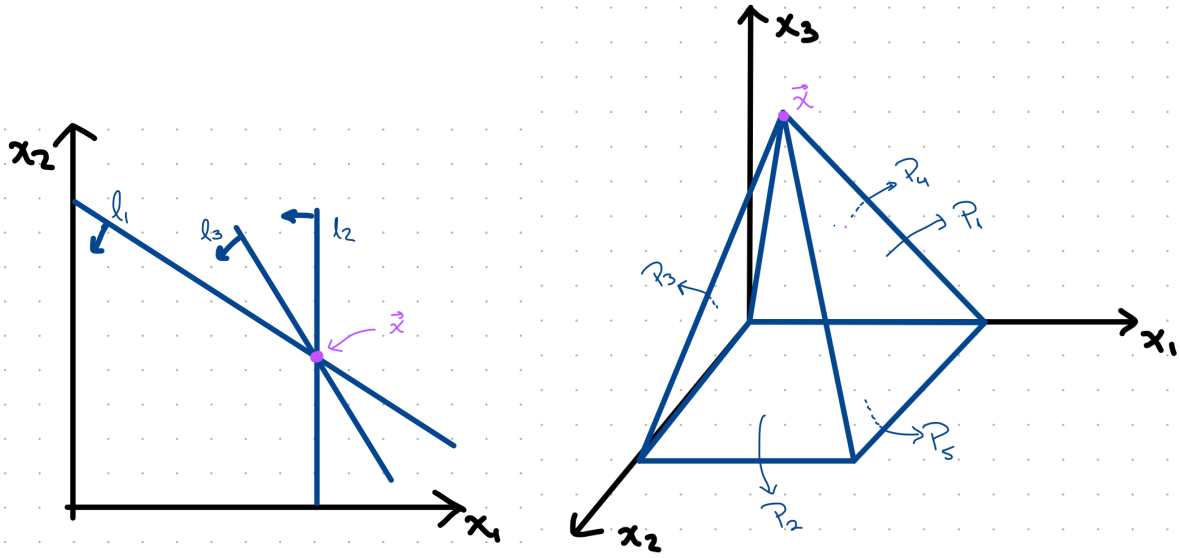
In the book you can see how this problem is solved using simplex. □

5.5 Degeneracy and zero-length simplex steps

When we developed simplex algorithm, we computed the step size for an improving simplex direction as the minimum number so that a basic variable becomes 0. At this point, we implicitly assumed that all the basic variables are nonzeros. However, this may not be the case. If a basic variable is 0, then we may get a step size $\lambda = 0$.

Definition 5.10. *A BFS to a standard-form linear program is degenerate if nonnegativity constraints for some basic variables are active, that is, if some basic variables have value 0.*

Degeneracy occurs when there are more active constraints than the minimum number needed to define a point. In general, if the feasible set is a subset of $\mathbb{R}^n$ (i.e., it is n-dimensional), we need n linear constraints to define a point. In the following figure we show two cases of degenerate extreme points.



In the left panel, we have a point $\boldsymbol{x}$ that is defined by the intersection of the three lines ℓ_1, ℓ_2, ℓ_3. Observe that in this case, the constraint ℓ_3 is redundant with ℓ_1 and ℓ_2, that is, the constraint ℓ_3 is not needed to define the feasible region.

In the right panel, we have a 3-dimensional feasible region and the point $\boldsymbol{x}$ is defined by the intersection of the four planes P_1, P_2, P_3, P_4. Even though the point $\boldsymbol{x}$ can be obtained as the intersection of any subset of 3 of these planes, none of the constraints defined by them is redundant.

The problem with degeneracy is that adjacent BFS result in the same extreme point because there is not a unique choice of basic variables to obtain the same extreme point.

Degeneracy can be a problem, since it may lead to simplex steps with no improvement of the objective function and, further, the algorithm may cycle. Giving steps with no improvement is not a big deal because simplex is still fast. Cycling can be an issue, but all we need to do to avoid it is tracking the extreme points we already evaluated and avoid repeating them.

5.6 Revised simplex

In the last section we saw that simplex may not improve in every step and, hence, we could reduce the number of steps we give to reach an optimal solution. However, this is not the main reason why simplex is not faster. A more efficient way to optimize simplex is by focusing on the computational time of each step.

Specifically, in each step we need to solve systems of equations to compute each of the simplex directions. In this section we describe the revised simplex algorithm, which makes use of matrix algebra to make each step more efficient.

Let B be a square matrix with the columns of A corresponding to basic variables, and N be the remaining columns of A, that is, the columns corresponding to nonbasic variables. Then, if we split the vector $\boldsymbol{x} = (\boldsymbol{x}_B, \boldsymbol{x}_N)$ where $\boldsymbol{x}_B$ represents the "subvector" with the basic variables and $\boldsymbol{x}_N = \mathbf{0}$ is the subvector of nonbasic variables, we can write:

$$A\boldsymbol{x} = B\boldsymbol{x}_B + N\boldsymbol{x}_N = B\boldsymbol{x}_B,$$

where the last equality holds because $\boldsymbol{x}_N = \mathbf{0}$ by definition of nonbasic variable. With this notation, we can compute a basic feasible solution as:

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

Additionally, when we compute the simplex directions we also use the basic columns of A to solve. Specifically, if we consider to go in the direction of the nonbasic variable x_j and we denote $A^{(j)}$ the column of A associated to the nonbasic variable x_j, we need to solve the following linear system of equations:

$$B\Delta\boldsymbol{x}_B = -A^{(j)} \qquad \Longrightarrow \qquad \Delta\boldsymbol{x}_B = -B^{-1}A^{(j)}$$

Hence, computing B^{-1} is essential to compute the current basic solution and all the simplex directions. Further, we only need to compute the inverse once and then we right-multiply by the corresponding vectors.

Before solving an example, let's remember how to compute the inverse of a 2×2 matrix.

Inverse of a 2×2 matrix
Consider a 2×2 matrix Z with the following elements:

$$Z = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Then, its inverse is

$$Z^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Let's see a short example.

Example 5.12 (Example 5.23 from the textbook)**.** *Consider the following standard-form linear program:*

$$\begin{aligned} \min \quad & 9x_1 + 3x_2 + x_4 \\ s.t. \quad & 2x_1 + x_2 - x_3 = 12 \\ & x_1 + 9x_3 + 2x_4 = 5 \\ & x_i \geq 0 \qquad \forall i \end{aligned}$$

and assume that x_1 and x_2 are the basic variables. Compute the current basic solution and all the simplex directions to move.

Solution. The matrix A and the vector $\boldsymbol{b}$ from the standard form are

$$A = \begin{bmatrix} 2 & 1 & -1 & 0 \\ 1 & 0 & 9 & 2 \end{bmatrix} \quad \text{and} \quad \boldsymbol{b} = \begin{pmatrix} 12 \\ 5 \end{pmatrix}$$

Then, the matrix B corresponding to the basic columns is

$$B = \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix}$$

and its inverse is

$$B^{-1} = \frac{1}{2 \times 0 - 1 \times 1} \begin{bmatrix} 0 & -1 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -2 \end{bmatrix}$$

Hence, the basic components of the current basic solution are

$$\boldsymbol{x}_B = B^{-1}\boldsymbol{b} = \begin{bmatrix} 0 & 1 \\ 1 & -2 \end{bmatrix} \begin{pmatrix} 12 \\ 5 \end{pmatrix} = \begin{pmatrix} 5 \\ 2 \end{pmatrix}$$

Hence, the current basic solution is

$$\boldsymbol{x} = (5, 2, 0, 0)$$

The nonbasic variables are x_3 and x_4. We first compute the simplex direction obtained by entering x_3 to the basis and we obtain

$$\Delta \boldsymbol{x}_B = -B^{-1}A^{(3)} = -\begin{bmatrix} 0 & 1 \\ 1 & -2 \end{bmatrix} \begin{pmatrix} -1 \\ 9 \end{pmatrix} = \begin{pmatrix} -9 \\ 19 \end{pmatrix}$$

Hence, the simplex direction to enter x_3 to the basis is

$$\Delta \boldsymbol{x} = (-9, 19, 1, 0)$$

To compute the simplex direction obtained by entering x_4 to the basis we do

$$\Delta \boldsymbol{x}_B = -B^{-1}A^{(4)} = -\begin{bmatrix} 0 & 1 \\ 1 & -2 \end{bmatrix}\begin{pmatrix} 0 \\ 2 \end{pmatrix} = \begin{pmatrix} -2 \\ 4 \end{pmatrix}$$

Hence, the simplex direction to enter x_3 to the basis is

$$\Delta \boldsymbol{x} = (-2, 4, 0, 1)$$

As you see, instead of solving three systems of equations, we compute the inverse of a matrix once and then only multiply. □

Updating B^{-1} efficiently

In the simplex algorithm, the basis changes in every iteration. Hence, the matrix B also changes. However, only one basic variables changes in every iteration, that is, the old matrix B and the new matrix B are equal, except by the column representing the variable that entered the basis. Here we learn how to make use of this information to update the matrix B efficiently.

Principle 5.13 (Principle 5.43 from the textbook)**.** *The new basis inverse at each iteration can be computed from the old by*

$$[\text{new } B^{-1}] = E[\text{old } B^{-1}]$$

where E is an update matrix constructed as follows. The columns of E equal the identity matrix, except by the column representing the variable that leaves the basis. If the basic variables are represented by the set $\overline{B}$ and the basic variable r leaves the basis, then the r^{th} column of E has elements

$$E_i^{(r)} = \begin{cases} -1/\Delta x_r & , \text{ if } i = r \\ -\Delta x_i/\Delta x_r & , \text{ if } i \neq r \end{cases} \qquad \forall i \in \overline{B} \tag{10}$$

Example 5.13 (Example 5.24 from the textbook)**.** *In the last example, assume that x_3 enters the basis and x_1 leaves. Update the matrix B^{-1} efficiently.*

Solution. Since x_3 enters the basis, we have $\Delta \boldsymbol{x} = (-9, 19, 1, 0)$. Then, since x_1 leaves the basis, we construct E by replacing the first column of the identity matrix as follows:

$$E = \begin{bmatrix} -1/\Delta x_1 & 0 \\ -\Delta x_2/\Delta x_1 & 1 \end{bmatrix} = \begin{bmatrix} 1/9 & 0 \\ 1/19 & 1 \end{bmatrix}$$

Hence, the new B^{-1} we have

$$\text{new } B^{-1} = E[\text{old } B^{-1}] = \begin{bmatrix} 1/9 & 0 \\ 1/19 & 1 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 1 & -2 \end{bmatrix} = \begin{bmatrix} 0 & 1/9 \\ 1 & 1/9 \end{bmatrix}$$

□

Computing the reduced costs efficiently

Another reason why the rudimentary simplex algorithm is not completely efficient, is that we need to compute all the simplex directions in order to decide if they are improving directions or not. Here we learn a more efficient way to compute the reduced costs. Then, we only compute the simplex direction for a single improving direction.

Let $\boldsymbol{c} = (\boldsymbol{c}_B, \boldsymbol{c}_N)$, where $\boldsymbol{c}_B$ represent the basic components of $\boldsymbol{c}$ and $\boldsymbol{c}_N$ the nonbasic ones. Suppose that we want to compute the reduced cost associated to the nonbasic variable x_j. Recall that the simplex direction of the nonbasic variable j satisfies:

$$\Delta \boldsymbol{x} = (\Delta \boldsymbol{x}_B, \Delta \boldsymbol{x}_N) \quad \text{, where } \Delta \boldsymbol{x}_B = -B^{-1}A^{(j)} \text{ and } \Delta \boldsymbol{x}_N = \boldsymbol{e}^{(j)}$$

Then, if we let $\overline{B}$ be the set of basic variables, by construction of the simplex directions we have

$$\overline{c}_j = \boldsymbol{c}^T \Delta \boldsymbol{x}$$

$$
\begin{aligned}
&= c_j + \sum_{i \in \overline{B}} c_i \Delta x_i \\
&= c_j + \boldsymbol{c}_B^T \Delta \boldsymbol{x}_B \\
&= c_j - \boldsymbol{c}_B^T B^{-1} A^{(j)} \qquad \text{(using that } \Delta \boldsymbol{x}_B \text{ in } j^{\text{th}} \text{ direction is } -B^{-1}A^{(j)})
\end{aligned}
$$

In the last equation, observe that c_j and $A^{(j)}$ depend on the nonbasic variable that we are evaluating to enter the basis, but $\boldsymbol{c}_B^T B^{-1}$ does not. Then, we only need to compute the vector-matrix multiplication $\boldsymbol{c}_B^T B^{-1}$ once to evaluate all the reduced costs.

Definition 5.11 (Definition 5.45 from the textbook)**.** *The pricing vector $\boldsymbol{v}$ corresponding to the current basis $\overline{B}$ with matrix B, is defined as*

$$\boldsymbol{v}^T \triangleq \boldsymbol{c}_B^T B^{-1}$$

Then, the reduced cost for every nonbasic variable j can be efficiently computed as:

$$\overline{c}_j = c_j - \boldsymbol{v}^T A^{(j)}$$

or, equivalently,

$$\overline{\boldsymbol{c}}_N = \boldsymbol{c}_N - N^T \boldsymbol{v}.$$

Observe that we are computing the reduced costs without computing the simplex directions, so this approach can save us a lot of time.

Example 5.14 (Example 5.26 from the textbook)**.** *Consider the following linear program*

$$
\begin{aligned}
\min \quad & 3x_1 + 100x_2 + 12x_3 - 8x_4 \\
s.t. \quad & 3x_1 + x_2 - x_3 = 90 \\
& -x_1 - x_2 + x_4 = 22 \\
& x_i \geq 0 \qquad \forall i \in \{1,2,3,4\}
\end{aligned}
$$

Assuming that the current basis is $\overline{B} = \{x_2, x_4\}$, compute:

(a) The corresponding basis inverse matrix

(b) The current basic solution

(c) The reduced costs on the nonbasic variables without generating any simplex directions. Are they improving directions?

(d) Choose an improving direction and compute the next basic feasible solution

(e) Update the matrix B^{-1} efficiently

(f) Verify that the matrix you computed is correct

Solution. The matrix A is given by

$$A = \begin{bmatrix} 3 & 1 & -1 & 0 \\ -1 & -1 & 0 & 1 \end{bmatrix}$$

Then, we obtain the following results:

(a) The basic variables are x_2, x_4, so the matrix B corresponds to the second and fourth columns of A, that is,

$$B = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \qquad \Longrightarrow \qquad B^{-1} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

(b) The basic components of the current basic solutions are:

$$\boldsymbol{x}_B = B^{-1}\boldsymbol{b} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{pmatrix} 90 \\ 22 \end{pmatrix} = \begin{pmatrix} 90 \\ 112 \end{pmatrix}$$

Hence, the current basic solution is $\boldsymbol{x}^{(0)} = (0, 90, 0, 112)$.

(c) We first compute the pricing vector:

$$\boldsymbol{v}^T = \boldsymbol{c}_B^T B^{-1} = (c_2 \; c_4)B^{-1} = (100 \;\; -8) \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = (92 \;\; -8)$$

Then, the reduced cost associated to x_1 and x_3 are, respectively:

$$\bar{c}_1 = c_1 - \boldsymbol{v}^T A^{(1)} = 3 - (92 \;\; -8) \begin{pmatrix} 3 \\ -1 \end{pmatrix} = 3 - 284 = -281$$

$$\bar{c}_3 = c_3 - \boldsymbol{v}^T A^{(3)} = 12 - (92 \;\; -8) \begin{pmatrix} -1 \\ 0 \end{pmatrix} = 12 + 92 = 104$$

Since this is a minimization problem, the simplex direction resulting from adding x_1 to the basis is improving and the direction resulting from adding x_3 is not improving.

(d) We choose the simplex direction of the nonbasic variable x_1. The first step to compute the next basic variable is to compute the simplex direction:

$$\Delta \boldsymbol{x}_B = -B^{-1}A^{(1)} = -\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{pmatrix} 3 \\ -1 \end{pmatrix} = \begin{pmatrix} -3 \\ -2 \end{pmatrix}$$

Then, the simplex direction is

$$\Delta \boldsymbol{x}^{(1)} = (1, -3, 0, -2)$$

because the nonbasic variables are x_1 and x_3, and we are computing the simplex direction associated to x_1.

Now that we have the simplex direction, we compute the step-size:

$$\begin{aligned} \lambda &= \min\left\{ \frac{x_2^{(0)}}{-\Delta x_2^{(1)}}, \; \frac{x_4^{(0)}}{-\Delta x_4^{(1)}} \right\} \\ &= \min\left\{ \frac{90}{-(-3)}, \; \frac{112}{-(-2)} \right\} \\ &= \min\{30, \; 56\} \\ &= 30 \end{aligned}$$

Then, the new basic variable is

$$\begin{aligned} \boldsymbol{x}^{(1)} &= \boldsymbol{x}^{(0)} + \lambda \Delta \boldsymbol{x}^{(1)} \\ &= (0, 90, 0, 112) + 30(1, -3, 0, -2) \\ &= (30, 0, 0, 52) \end{aligned}$$

(e) When we updated the basic solution, the variable x_1 entered the basis and the variable x_2 left the basis. The current basis is $\overline{B} = \{x_2, x_4\}$. Then, the column of B associated to the basic variable x_2 is the first column. Then, we modify the first column of the matrix E and obtain

$$E = \begin{bmatrix} -1/\Delta x_2^{(1)} & 0 \\ -\Delta x_4^{(1)}/\Delta x_2^{(1)} & 1 \end{bmatrix} = \begin{bmatrix} 1/3 & 0 \\ -2/3 & 1 \end{bmatrix}$$

Therefore,

$$\text{new } B^{-1} = \begin{bmatrix} 1/3 & 0 \\ -2/3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1/3 & 0 \\ 1/3 & 1 \end{bmatrix}$$

(f) To verify that the matrix we obtained is correct, we compute it from scratch. That is, the new basis is $\overline{B} = \{x_1, x_4\}$. Then, the new matrix B corresponds to the columns 1 and 4 from the matrix A. We obtain

$$\text{new } B = \begin{bmatrix} 3 & 0 \\ -1 & 1 \end{bmatrix} \qquad \Longrightarrow \qquad \text{new } B^{-1} = \begin{bmatrix} 1/3 & 0 \\ 1/3 & 1 \end{bmatrix}$$

□

To close the chapter, we write the revised simplex algorithm explicitly.

Algorithm 5.2 (Revised simplex algorithm)**.**

0. ***Initialization***

0.1 Choose any starting feasible basis, compute the corresponding matrix B and its inverse B^{-1}

0.2 Compute the corresponding basic feasible solution $\boldsymbol{x}^{(0)} = B^{-1}\boldsymbol{b}$ and set $t \leftarrow 0$

1. ***Reduced costs***

1.1 Compute the pricing vector $\boldsymbol{v} = B^{-1}\boldsymbol{c}_B$

1.2 For each nonbasic variable j, compute the reduced cost $\overline{c}_j = c_j - \boldsymbol{v}^T A^{(j)}$

2. ***Optimality***

- *If no direction is improving, then* ***STOP. Current solution $\boldsymbol{x}^{(t)}$ is optimal***
- *Otherwise, choose any improving direction p and compute the associated simplex direction $\Delta\boldsymbol{x}_B^{(t+1)} = -B^{-1}A^{(p)}$, $\Delta\boldsymbol{x}_N^{(t+1)} = \boldsymbol{e}^{(p)}$*

3. ***Step size***

- *If $\Delta\boldsymbol{x}^{(t+1)} > \boldsymbol{0}$, that is, if all the elements of the improving direction are strictly positive, then* ***STOP. The given model is unbounded.***
- *Otherwise, choose the variable that leaves the basis x_r so that:*

$$\frac{x_r^{(t)}}{-\Delta x_r^{(t+1)}} = \min\left\{ \frac{x_j^{(t)}}{-\Delta x_j^{(t+1)}} : \ \Delta x_j^{(t+1)} < 0 \right\}$$

and set

$$\lambda \leftarrow \frac{x_r^{(t)}}{-\Delta x_r^{(t+1)}}$$

4. ***Step 4: New point and basis***

4.1 Compute the new solution

$$\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} + \lambda\Delta\boldsymbol{x}^{(t+1)}$$

4.2 Update B^{-1} by left-multiplying by the matrix E using Equation (10)

4.3 Advance $t \leftarrow t + 1$

4.4 Return to step 1.

6 Sensitivity Analysis, Duality and Optimality

[This chapter is based on Chapter 6 of the textbook]

So far we have been solving linear programs with fixed parameters, that is, fixed objective-function weights, fixed weights on the left-hand side of the constraints and fixed right-hand sides. However, in real life we usually do not know exactly what are the parameters of the problems, so we use estimates. For example, if our problem involves the demand for a product, we may have a forecast but we do not know the exact future demand.

In this chapter we will learn a systematic method to perform sensitivity analysis, that is, to analyze how would our optimal solutions change if we change some of the parameters of the problem. We will see that sensitivity analysis on the cost vector $\boldsymbol{c}$ is substantially different from sensitivity analysis on the constraint parameters. However, we will see that both types of sensitivity analysis are related and, indeed, we can construct an equivalent linear problem such that the objective function becomes the right-hand side vector and viceversa. Such problem is called the **dual**.

Let's start with a generic interpretation of each part of a typical linear program.

6.1 Generic interpretation of LP's

As we have seen along the semester, each optimization problem is a whole new world in terms of the meaning of the variables, objective function, and constraints. However, there are some common factors that we will use in this chapter to make sense of the sensitivity analysis methods we will learn.

Principle 6.1 (based on Principles 6.1-6.8 from the textbook)**.** *In a generic optimization problem, we have the following interpretations:*

- ***Objective function:*** *Minimize cost or Maximize benefits*
- ***Main constraints:***
  - $\leq$ *constraints typically represent limited supply*
  - $\geq$ *constraints typically represent demand satisfaction*
  - $=$ *constraints mean that the supply and demand must be satisfied exactly*
- ***Decision variables:*** *Level of activity*

Hence, the ***left-hand side coefficients*** *represent the impact per unit of activity of each variable on the available resources.*

6.2 Qualitative sensitivity analysis

In this section we intuitively analyze what are the effects of modifying the constraints, objective function and number of variables in an optimization problem. We start with the constraints.

Effects of modifying constraints

We start with a definition.

Definition 6.1. *We say that we relax/tighten a constraint when we modify a constraint such that the feasible region becomes larger/smaller.*

In other words, when we relax a constraint we are adding more solutions, and when we tighten it, we are removing some constraints. Under this interpretation, intuitively, the following principle holds.

Naturally, by definition of relaxing and tightening an optimization problem, dropping constraints relaxes the problem and adding more constraints will tighten it.

Principle 6.2 (Principle 6.9 from the textbook)**.** *Relaxing the constraints either leaves the optimal value unchanged or makes it better. Tightening the constraints either leaves the optimal value unchanged or makes it worse.*

Observe that the only constraints that we can relax or tighten are the inequality constraints. In the following principle we establish how to do it if we only want to modify the right-hand side parameters.

Principle 6.3 (based on Principle 6.10 from the textbook)**.** *Changes on the right-hand side parameters of the constraints affects the optimization problems as follows:*

Constraint type	*RHS increase*	*RHS decrease*
Supply $\leq$	*Relax*	*Tighten*
Demand $\geq$	*Tighten*	*Relax*

Let's see a short example.

Example 6.1. *Consider the feasible region defined by* $\mathcal{P} = \{(x_1, x_2) \in \mathbb{R}^2_+ : \ x_1 + x_2 \geq 1, \ x_1 + 3x_2 \leq 9, \ x_1 \leq 3\}$. *Plot the feasible region along with the effects of the following changes to the right-hand side of the constraints and identify which changes correspond to relaxing and tightening the constraints:*

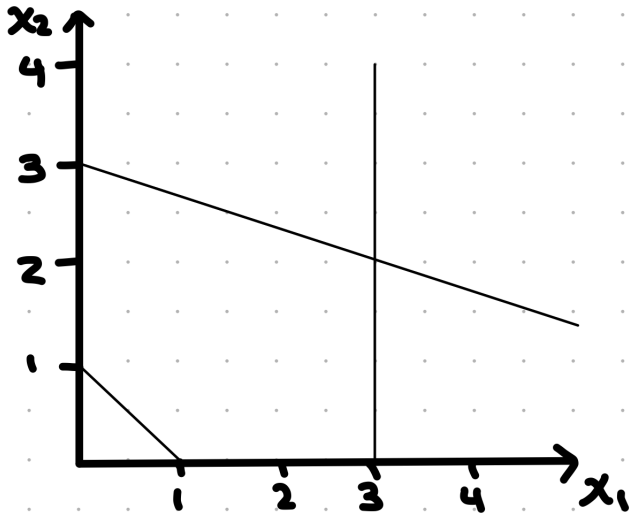
(a) Modify the first constraint to $x_1 + x_2 \geq 2$

(b) Modify the first constraint to $x_1 + x_2 \geq 1/2$

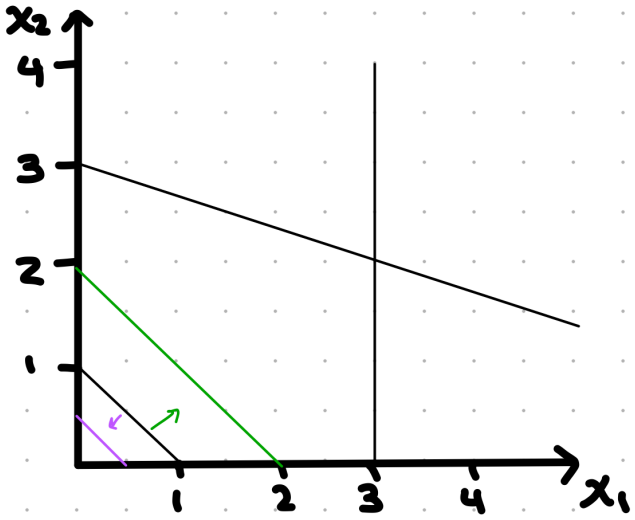
(c) Modify the second constraint to $x_1 + 3x_2 \leq 12$

(d) Modify the second constraint to $x_1 + 3x_2 \leq 6$

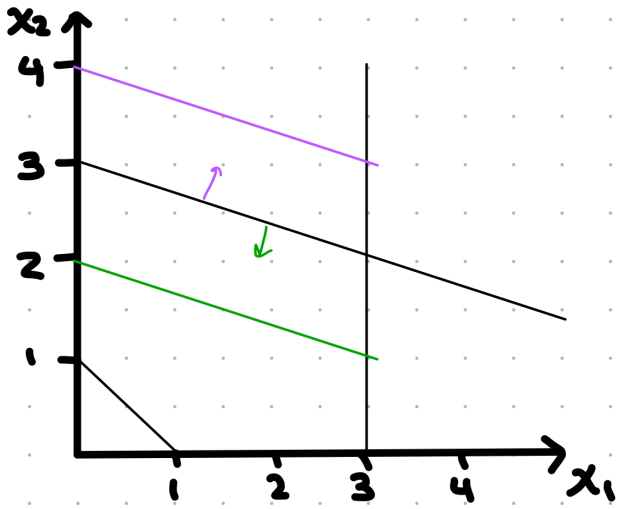
Solution. The feasible region is



The modifications (a) and (b) are plotted below:



Observe that (a) tightens the constraint and (b) relaxes the constraint.
The modifications (c) and (d) are plotted below:



Observe that (c) relaxes the constraint and (d) tightens the constraint.

□

We may as well change the coefficients of the variables on the left-hand side of the constraints. We will assume that all the constraints have a nonnegative right-hand side from now on. If this is not true, we can always rewrite the constraint by multiplying everything by -1. We obtain the following principle:

Principle 6.4 (based on Principle 6.11 from the textbook)**.** *Changes on the left-hand side coefficients of a constraint with nonnegative right-hand side has the following effects:*

Constraint type	*LHS coef. increase*	*LHS coef. decrease*
Supply $\leq$	*Tighten*	*Relax*
Demand $\geq$	*Relax*	*Tighten*

In this case, the graphical effect on the constraints will be related to their slope. Let's continue working on our example.

Example 6.2. *Consider the feasible region from Example 6.1. Plot the following modifications to the left-hand side parameters and identify which changes correspond to relaxing and tightening the constraints:*

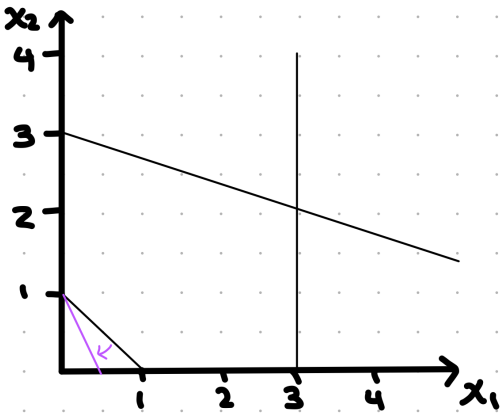
(e) Modify the first constraint to $2x_1 + x_2 \geq 1$

(f) Modify the first constraint to $(1/2)x_1 + x_2 \geq 1$

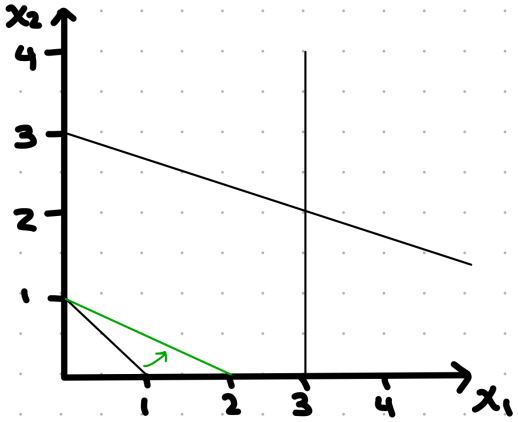
(g) Modify the second constraint to $x_1 + 4x_2 \leq 9$

(h) Modify the second constraint to $x_1 + 2x_2 \leq 9$

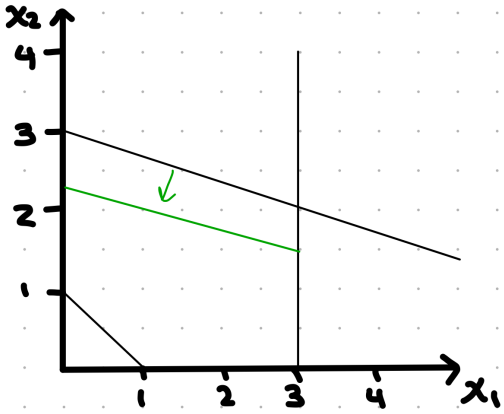
Solution. We obtain the following changes in the feasible region, where pink represents relaxing and green represents tightening.



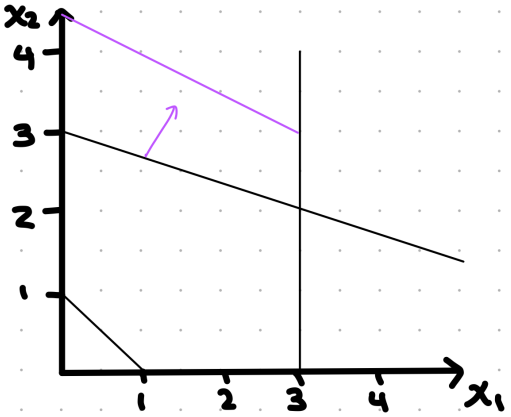
(e)



(f)



(g)



(h)

□

Effects of modifying the objective function coefficients

Naturally, we may also modify the coefficients of the variables in the objective function. This modifications have nothing to do with the feasible region and, hence, they do not relax nor tighten the problem. However, we can still easily predict if the optimal value will be better or worse after modifying the weight of nonnegative variables.

Intuitively, if we are maximizing, we would like that the same optimal solution gives us a higher objective function value. Hence, in a maximizing problem, increasing the objective-function coefficient of a nonnegative variable should improve the optimal value. Similarly, in a minimization problem, decreasing the objective-function coefficient should improve the optimal value. We establish these properties in the following principle.

Principle 6.5 (based on Principle 6.15 from the textbook)**.** *Changing the objective function coefficient of a nonnegative variable in an optimization model affects the optimal value as follows:*

Objective function type	*Coefficient increase*	*Coefficient decrease*
Maximize	*Same or better*	*Same or worse*
Minimize	*Same or worse*	*Same or better*

Effects of adding or dropping variables

Recall that each variable represents a choice that we need to make. The more choices, the better. Hence, we obtain the following principle.

Principle 6.6 (based on Principle 6.17 from the textbook)**.** *Adding variables must leave the optimal value unchanged or improved, and dropping activities will leave the value unchanged or degraded.*

6.3 Quantitative analysis: Duality

In the previous section we intuitively analyzed if certain changes in the parameters will help or hurt the objective function value. In this section we will learn to quantify these changes. Specifically, we will learn how to formulate a linear program that is related to the problem we are solving in order to understand these changes. We start with some definitions.

Definition 6.2 (Definitions 6.18 and 6.19 from the textbook)**.**

(i) The primal is the given optimization model, the one formulating the application of primary interest.

(ii) The dual corresponding to a given primal formulation is a closely related LP with decision variables and constraints that quantify the sensitivity of primal results to changes in its parameters.

So far, we have only worked with primal optimization models. The dual is indeed an optimization problem with the same parameters as the primal, that is, it also depends on $A, \boldsymbol{c}, \boldsymbol{b}$. However, the decision variables of the primal correspond to constraints on the dual and viceversa. Indeed, the objective-function cost vector of the dual is $\boldsymbol{b}$ and the right-hand side vector is $\boldsymbol{c}$. We will shortly see specifically how to construct the dual.

Principle 6.7 (based on Principles 6.20 and 6.21 from the dual)**.** *There is a decision variable in the dual for each main constraint in the primal. We denote ν_i the dual decision variable associated to the i^{th} primal main constraint, and the sign restriction of ν_i depends on whether the primal optimal value improves or not when we change the corresponding right-hand side. Specifically, we obtain the following relationship:*

Type of constraint	*Minimize objective*	*Maximize objective*
Constraint i is $\leq$	$\nu_i \leq 0$	$\nu_i \geq 0$
Constraint i is $\geq$	$\nu_i \geq 0$	$\nu_i \leq 0$
Constraint i is $=$	ν_i *is unrestricted*	ν_i *is unrestricted*

In the principle above, observe the relationship to the intuition we built when modifying the right-hand side of each constraint:

- In a supply constraint ($\leq$), an increase on the right-hand side improves the objective function. In a minimization problem, better means that the objective function decreases, and we obtain $\nu_i \leq 0$. In a maximization problem, the objective function increases and, hence, we get $\nu_i \geq 0$.
- In a demand constraint ($\geq$), an increase on the right-hand side makes the objective function worse. In a minimization problem, worse means that the objective function increases and, hence, we get $\nu_i \geq 0$. In the maximization problem, worse implies that the objective function value decreases and, therefore, we get $\nu_i \leq 0$
- Equality constraints act like both, supply and demand constraints. Therefore, the corresponding dual variable can be either positive or negative, but we don't know before solving.

In conclusion, if increasing the right-hand side of a constraint i makes the objective function value increase, then the associated dual variable $\nu_i \geq 0$; and if increasing the right-hand side of a constraint i makes the objective function value decrease, then the associated dual variable $\nu_i \leq 0$.

We just learned that the sign of the dual variables indicates whether the objective function value increases or decreases. Indeed, the value of the dual variables in the optimal solution indicates how much the objective function will change per unit of change on the corresponding right-hand side. Formally, we have the following principle.

Principle 6.8 (based on Principle 6.22 from the textbook)**.** *Dual variables provide implicit prices for the marginal unit of the resource modeled by each constraint as its right-hand side limit is encountered.*

In other words, the dual variables price the resources. Hence, we get the following principle.

Principle 6.9 (based on Principle 6.25 from the textbook)**.** *If a primal linear program has an optimal solution $\boldsymbol{x}^*$, then the dual also has an optimal solution $\boldsymbol{\nu}^*$. Further, the optimal value of the primal and the dual is the same, that is,*

$$\sum_j c_j x_j^* = \sum_i b_i \nu_i^*$$

The above property is extremely important, and it is specific to linear programs. It is called **strong duality**.

6.4 Formulating linear programming duals

In the following table we provide the formula to compute the dual.

Primal	Dual
Objective: max $\boldsymbol{c}^T\boldsymbol{x}$	Objective: min $\boldsymbol{b}^T\boldsymbol{\nu}$
Constraint: $\sum_j a_{i,j}x_j \geq b_i$	Variable: $\nu_i \leq 0$
Constraint: $\sum_j a_{i,j}x_j = b_i$	Variable: ν_i unrestricted
Constraint: $\sum_j a_{i,j}x_j \leq b_i$	Variable: $\nu_i \geq 0$
Variable: $x_j \geq 0$	Constraint: $\sum_i a_{i,j}\nu_i \geq c_j$
Variable: x_j unrestricted	Constraint: $\sum_i a_{i,j}\nu_i = c_j$
Variable: $x_j \leq 0$	Constraint: $\sum_i a_{i,j}\nu_i \leq c_j$
Objective: min $\boldsymbol{c}^T\boldsymbol{x}$	Objective: max $\boldsymbol{b}^T\boldsymbol{\nu}$
Constraint: $\sum_j a_{i,j}x_j \geq b_i$	Variable: $\nu_i \geq 0$
Constraint: $\sum_j a_{i,j}x_j = b_i$	Variable: ν_i unrestricted
Constraint: $\sum_j a_{i,j}x_j \leq b_i$	Variable: $\nu_i \leq 0$
Variable: $x_j \geq 0$	Constraint: $\sum_i a_{i,j}\nu_i \leq c_j$
Variable: x_j unrestricted	Constraint: $\sum_i a_{i,j}\nu_i = c_j$
Variable: $x_j \leq 0$	Constraint: $\sum_i a_{i,j}\nu_i \geq c_j$

Observe that in the constraints of the dual, the sum is over i. That is, to compute them we look at the appearances of each x_j in each of the constraints and we write their coefficients. To exemplify this fact, we show how to compute the dual in primals with nonnegative variables. We obtain the following correspondence:

$$\begin{array}{c|c} \text{Primal} & \text{Dual} \\ \begin{array}{ll} \max & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.} & A\boldsymbol{x} \begin{bmatrix} \leq \\ \geq \\ = \end{bmatrix} \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{array} & \begin{array}{ll} \min & \boldsymbol{b}^T\boldsymbol{\nu} \\ \text{s.t.} & A^T\boldsymbol{\nu} \geq \boldsymbol{c} \\ & \boldsymbol{\nu} \begin{bmatrix} \geq \\ \leq \\ \text{unrestricted} \end{bmatrix} \boldsymbol{0} \end{array} \end{array}$$

Then, each constraint of the primal is associated to a variable in the dual; and each constraint of the dual is associated to a variable of the primal.

Let's do some examples.

Example 6.3 (based on Examples 6.17 and 6.18 of the textbook)**.** *For the following linear optimization problems, compute the dual.*

(a)
$$\begin{array}{ll} \min & 30x_1 + 5x_3 \\ s.t. & x_1 - x_2 + x_3 \geq 1 \\ & 3x_1 + x_2 = 4 \\ & 4x_2 + x_3 \leq 10 \\ & x_j \geq 0 \qquad \forall j \end{array}$$

(b)
$$\begin{array}{ll} \max & 6x_1 - x_2 + 13x_3 \\ s.t. & 3x_1 + x_2 + 2x_3 = 7 \\ & 5x_1 - x_2 \leq 6 \\ & x_2 + x_3 \geq 2 \\ & x_1 \geq 0,\ x_2 \leq 0 \end{array}$$

(c)
$$\begin{array}{ll} \min & 7x_1 + 44x_3 \\ s.t. & -2x_1 - 4x_2 + x_3 \leq 15 \\ & x_1 + 4x_2 \geq 5 \\ & 5x_1 - x_2 + 3x_3 = -11 \\ & x_1 \leq 0,\ x_3 \geq 0 \end{array}$$

Solution. Following the rules from the table above, we obtain:

(a) $$\begin{aligned} \max \quad & \nu_1 + 4\nu_2 + 10\nu_3 \\ \text{s.t.} \quad & \nu_1 + 3\nu_2 \le 30 \\ & -\nu_1 + \nu_2 + 4\nu_3 \le 0 \\ & \nu_1 + \nu_3 \le 5 \\ & \nu_1 \ge 0,\ \nu_2 \text{ ur},\ \nu_3 \le 0 \end{aligned}$$

(b) $$\begin{aligned} \min \quad & 7\nu_1 + 6\nu_2 + 2\nu_3 \\ \text{s.t.} \quad & 3\nu_1 + 5\nu_2 \ge 6 \\ & \nu_1 - \nu_2 + \nu_3 \le -1 \\ & 2\nu_1 + \nu_3 = 13 \\ & \nu_1 \text{ ur},\ \nu_2 \ge 0,\ \nu_3 \le 0 \end{aligned}$$

(c) $$\begin{aligned} \max \quad & 15\nu_1 + 5\nu_2 - 11\nu_3 \\ \text{s.t.} \quad & -2\nu_1 + \nu_2 + 5\nu_3 \ge 7 \\ & -4\nu_1 + 4\nu_2 - \nu_3 = 0 \\ & \nu_1 + 3\nu_3 \le 44 \\ & \nu_1 \le 0,\ \nu_2 \ge 0,\ \nu_3 \text{ ur} \end{aligned}$$

□

6.5 Duality and Optimality in Linear Programming

We started studying duality as a method to quantify sensitivity analysis. However, the dual problem is much more powerful than a sensitivity analysis tool. In this section we will explore how duality relates to the primal optimal solution.

Let's first agree on the notation. For simplicity, let's consider the following primal and its dual:

$$(P) \quad \begin{aligned} \min \quad & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t} \quad & A\boldsymbol{x} \ge \boldsymbol{b} \\ & \boldsymbol{x} \ge \boldsymbol{0} \end{aligned} \qquad\qquad (D) \quad \begin{aligned} \max \quad & \boldsymbol{b}^T\boldsymbol{\nu} \\ \text{s.t.} \quad & A^T\boldsymbol{\nu} \le \boldsymbol{c} \\ & \boldsymbol{\nu} \ge \boldsymbol{0} \end{aligned}$$

Observe that the primal may have $\le$ and $=$ constraints too, and the variables may not be all nonnegative. However, we can always rewrite our primal to match (P). So, without loss of generality, in this section we work with the primal (P) and its dual (D).

The first property that we will formally state is that computing duals is a symmetric task.

Principle 6.10 (Principle 6.46 from the textbook)**.** *The dual of the dual of any linear program is the primal.*

Example 6.4. *For the last primal in Example 6.3, verify that the dual of the dual is the primal.*

Solution. We had the following primal and dual:

$$(P-c) \quad \begin{aligned} \min \quad & 7x_1 + 44x_3 \\ \text{s.t.} \quad & -2x_1 - 4x_2 + x_3 \le 15 \\ & x_1 + 4x_2 \ge 5 \\ & 5x_1 - x_2 + 3x_3 = -11 \\ & x_1 \le 0,\ x_3 \ge 0 \end{aligned} \qquad\qquad (D-c) \quad \begin{aligned} \max \quad & 15\nu_1 + 5\nu_2 - 11\nu_3 \\ \text{s.t.} \quad & -2\nu_1 + \nu_2 + 5\nu_3 \ge 7 \\ & -4\nu_1 + 4\nu_2 - \nu_3 = 0 \\ & \nu_1 + 3\nu_3 \le 44 \\ & \nu_1 \le 0,\ \nu_2 \ge 0 \end{aligned}$$

Now we compute the dual of $(D-c)$ and we obtain

$$\begin{aligned} \min \quad & 7x_1 + 44x_3 \\ \text{s.t.} \quad & -2x_1 - 4x_2 + x_3 \le 15 \\ & x_2 + 4x_3 \ge 5 \\ & 5x_1 - x_3 + 3x_3 = -11 \\ & x_1 \le 0,\ x_3 \ge 0 \end{aligned}$$

which exactly equal to $(P-c)$. □

Weak duality

The next property relates **any** primal feasible solution with **any** dual feasible solution. It is called **weak duality**.

Principle 6.11 (Principle 6.47 from the textbook)**.** *The primal objective function evaluated at any feasible solution to a minimize primal is always an upper bound to the objective function value of the corresponding dual at any dual feasible solution. Formally, if $\overline{\boldsymbol{x}}$ is a primal feasible solution and $\overline{\boldsymbol{\nu}}$ is a dual feasible solution, we have*

$$\boldsymbol{c}^T\overline{\boldsymbol{x}} \geq \boldsymbol{b}^T\overline{\boldsymbol{\nu}}$$

Let's prove this result.

Proof. We will show that $\boldsymbol{c}^T\overline{\boldsymbol{x}} - \boldsymbol{b}^T\overline{\boldsymbol{\nu}}$ is nonnegative. Recall

$$\begin{array}{llll} (P) & \min & \boldsymbol{c}^T\boldsymbol{x} & \\ & \text{s.t} & A\boldsymbol{x} \geq \boldsymbol{b} & \\ & & \boldsymbol{x} \geq \boldsymbol{0} & \end{array} \qquad \begin{array}{llll} (D) & \max & \boldsymbol{b}^T\boldsymbol{\nu} & \\ & \text{s.t.} & A^T\boldsymbol{\nu} \leq \boldsymbol{c} & \\ & & \boldsymbol{\nu} \geq \boldsymbol{0} & \end{array}$$

We want to use that $\overline{\boldsymbol{x}}$ is primal feasible and $\overline{\boldsymbol{\nu}}$ is dual feasible, so we add and subtract a term that will allow us to use this property. We obtain

$$\begin{aligned} \boldsymbol{c}^T\overline{\boldsymbol{x}} - \boldsymbol{b}^T\overline{\boldsymbol{\nu}} &= \boldsymbol{c}^T\overline{\boldsymbol{x}} - \overline{\boldsymbol{\nu}}^T A\overline{\boldsymbol{x}} + \overline{\boldsymbol{\nu}}^T A\overline{\boldsymbol{x}} - \boldsymbol{b}^T\overline{\boldsymbol{\nu}} \\ &= \overline{\boldsymbol{x}}^T\boldsymbol{c} - \overline{\boldsymbol{x}}^T A^T\overline{\boldsymbol{\nu}} + \overline{\boldsymbol{\nu}}^T A\overline{\boldsymbol{x}} - \overline{\boldsymbol{\nu}}^T\boldsymbol{b} \qquad \text{(reorganizing terms)} \\ &= \overline{\boldsymbol{x}}^T\left(\boldsymbol{c} - A^T\overline{\boldsymbol{\nu}}\right) + \overline{\boldsymbol{\nu}}^T\left(A\overline{\boldsymbol{x}} - \boldsymbol{b}\right) \\ &\geq 0 \end{aligned}$$

where the last inequality holds because:

- $\overline{\boldsymbol{x}} \geq \boldsymbol{0}$ and $A\overline{\boldsymbol{x}} - \boldsymbol{b} \geq \boldsymbol{0}$ because $\overline{\boldsymbol{x}}$ is primal feasible
- $\overline{\boldsymbol{\nu}} \geq \boldsymbol{0}$ and $\boldsymbol{c} - A^T\overline{\boldsymbol{\nu}} \geq \boldsymbol{0}$ because $\overline{\boldsymbol{\nu}}$ is dual feasible.

This completes the proof. □

Earlier we learned the strong duality property, that said that the optimal value of the primal and the dual were the same. Here we simply establish that the primal is always an upper bound to the dual. Even though weak duality is not as powerful as strong duality, it still helps us to predict the relationships presented in the following principle.

Principle 6.12 (Principle 6.49)**.** *Consider a minimize primal and a maximize dual. Then, the possible outcomes for pairs of primal/dual can be summarized as follows:*

		Dual		
		Optimal	**Infeasible**	**Unbounded**
Primal	**Optimal**	*Possible*	*Never*	*Never*
	Infeasible	*Never*	*Possible*	*Possible*
	Unbounded	*Never*	*Possible*	*Never*

Let's analyze a bit deeper how to compute the table.

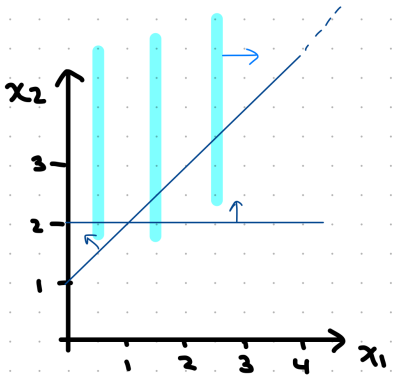
- If the primal has an optimal solution, then the dual also must have one (and viceversa). Then, the only possible outcome in the first row and first column is: primal and dual optimal.
- Since the primal is a minimization, an infeasible outcome means its optimal value is ∞. Then, since the dual is a maximization and weak duality indicates that it is a lower bound, the dual can be infeasible or unbounded.
- If the primal is unbounded, its optimal value must be $-\infty$ (because it is a minimization problem). Weak duality says that the dual must be a lower bound and, hence, it must also have $-\infty$ as optimal value. Since the dual is a maximization, the only possibility is that it is infeasible.

Let's do an example.

Example 6.5. *Show graphically that the following linear program is unbounded, and verify the implication for its dual.*

$$\begin{aligned} \max \quad & x_1 \\ s.t. \quad & -x_1 + x_2 \geq 1 \\ & x_2 \geq 2 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Solution. To solve graphically, observe that the objective function is x_1, so the level curves are vertical lines and the objective function value grows to the right. We have



Hence, the problem is unbounded because there is no upper bound for x_1.
The dual is

$$\begin{aligned} \min \quad & \nu_1 + \nu_2 \\ \text{s.t.} \quad & -\nu_1 \geq 1 \\ & \nu_1 + \nu_2 \geq 0 \\ & \nu_1, \nu_2 \leq 0 \end{aligned}$$

and it is infeasible because both variables are nonpositive and $\nu_1 \leq -1$. Hence, the second constraint cannot be satisfied. □

In practice, the number of variables and constraints that a model have can be huge. Then, solving to optimality can be challenging. However, finding a feasible solution is not hard. Weak duality tells us that if we have a primal and its dual, the difference between the min and the max is always positive. Then, we can compute feasible solutions and evaluate the duality gap, that is, the difference between the primal and dual objective function values. Let's think of a minimizing primal and a maximizing dual, then, the duality gap is

$$\text{Duality gap} = \boldsymbol{c}^T \overline{\boldsymbol{x}} - \boldsymbol{b}^T \overline{\boldsymbol{\nu}},$$

where $\overline{\boldsymbol{x}}$ and $\overline{\boldsymbol{\nu}}$ represent a primal and a dual feasible solution, respectively. When the duality gap is small, we know we are close to the optimal and, depending on how much resources we have (time or computer power), we might want to stop at a suboptimal solution that is "close enough."

Complementary slackness

Weak duality gives us a method to learn the outcome (optimal, infeasible or unbounded) of the dual depending on the outcome of the primal, and strong duality says that, whenever they are optimal, they have the same optimal value. However, none of these properties allow us to compute the dual optimal solution in terms of the primal, or viceversa. We learn how to do these computations with the following property, called **complementary slackness**.

Principle 6.13 (Principle 6.50 from the textbook)**.** *Suppose $\overline{\boldsymbol{x}}$/$\overline{\boldsymbol{\nu}}$ is a feasible solution to the primal/dual linear program.*

(i) If the objective values agree ($\boldsymbol{c}^T\overline{\boldsymbol{x}} = \boldsymbol{b}^T\overline{\boldsymbol{\nu}}$), then both $\overline{\boldsymbol{x}}$ and $\overline{\boldsymbol{\nu}}$ are optimal solutions and the following conditions are satisfied:

$$\left(c_j - \sum_i a_{ij}\overline{\nu}_i\right)\overline{x}_j = 0 \qquad \forall j \qquad\qquad \left(b_i - \sum_j a_{ij}\overline{x}_j\right)\overline{\nu}_i = 0 \qquad \forall i$$

(ii) If the objective values do not agree, then the difference of their objective values $\boldsymbol{c}^T\overline{\boldsymbol{x}} - \boldsymbol{b}^T\overline{\boldsymbol{\nu}}$ is exactly the total complementary slackness between the solutions, that is,

$$\boldsymbol{c}^T\overline{\boldsymbol{x}} - \boldsymbol{b}^T\overline{\boldsymbol{\nu}} = \sum_{j=1}^{n}\left(c_j - \sum_{i=1}^{m} a_{ij}\overline{\nu}_i\right)\overline{x}_j + \sum_{i=1}^{m}\left(\sum_j a_{ij}\overline{x}_j - b_i\right)\overline{\nu}_i$$

The conditions (i) are saying that for each primal variable $\overline{x}_j$, either the slack of the primal nonnegativity constraint or the slack of the corresponding dual constraint must be zero for the optimal solution. Similarly, for each dual variable $\overline{\nu}_i$, either the slack of the nonnegativity constraint or the slack of the corresponding primal constraint must be zero.

Let's focus on the primal complementary slackness (second set of equalities above) and develop an intuition of why it should be true. Recall that ν_i represents how much the optimal value would change if the right-hand side of the i^{th} constraint was increased 1 unit. Then,

- If $b_i - \sum_j a_{i,j}x_j \neq 0$, then the i^{th} constraint is inactive. That is, if we change a little bit the right-hand side, the constraint will remain inactive and, hence, the optimal value should not change. Therefore, if $b_i - \sum_j a_{i,j}x_j \neq 0$, then $\nu_i = 0$
- If $\nu_i \neq 0$, then we expect the optimal value to change even if we make small changes on the right-hand side of the i^{th} equation. Also, we know that modifying inactive constraints does not change the optimal solution, so the i^{th} constraint must be active. That is, it must be satisfied at equality and $b_i - \sum_j a_{i,j}x_j = 0$

In conditions (ii) we are trying to compute 'how far away' the current solution $\overline{\boldsymbol{x}}, \overline{\boldsymbol{\nu}}$ is from optimality. Then, we need to make sure that all the elements in the sum are positive. We wrote the above assuming that the primal constraints are $A\boldsymbol{x} \geq \boldsymbol{b}$, $\boldsymbol{x} \geq \boldsymbol{0}$ and the dual constraints are $A^T\boldsymbol{\nu} \leq \boldsymbol{c}$, $\boldsymbol{\nu} \geq \boldsymbol{0}$. A more general way to write the equality is:

$$\boldsymbol{c}^T\overline{\boldsymbol{x}} - \boldsymbol{b}^T\overline{\boldsymbol{\nu}} = \sum_{j=1}^{n}\left|c_j - \sum_{i=1}^{m} a_{ij}\overline{\nu}_i\right||\overline{x}_j| + \sum_{i=1}^{m}\left|\sum_j a_{ij}\overline{x}_j - b_i\right||\overline{\nu}_i|$$

Let's do an example.

Example 6.6. *Consider the following linear program, which has optimal solution $\boldsymbol{x}^* = (0, 5, 3)$.*

$$\begin{aligned} \max \quad & 5x_1 + 7x_2 + 10x_3 \\ s.t. \quad & 2x_1 - x_2 + 5x_3 \leq 10 \\ & x_1 + 3x_2 \leq 15 \\ & x_j \geq 0 \qquad \forall j \end{aligned}$$

Compute the dual program and a dual optimal solution.

Solution. We obtain the following dual problem:

$$\begin{aligned} \min \quad & 10\nu_1 + 15\nu_2 \\ \text{s.t.} \quad & 2\nu_1 + \nu_2 \geq 5 \\ & -\nu_1 + 3\nu_2 \geq 7 \\ & 5\nu_1 \geq 10 \\ & \nu_1, \nu_2 \geq 0 \end{aligned}$$

To compute the dual optimal solution we use complementary slackness. We obtain the following system of equations:

$$
\begin{aligned}
&(10 - 2x_1^* + x_2^* - 5x_3^*)\nu_1^* = 0\\
&(15 - x_1^* - 3x_2^*)\nu_2^* = 0\\
&(5 - 2\nu_1^* - \nu_2^*)x_1^* = 0\\
&(7 + \nu_1^* - 3\nu_2^*)x_2^* = 0\\
&(10 - 5\nu_1^*)x_3^* = 0
\end{aligned}
$$

Using that $\boldsymbol{x}^* = (0, 5, 3)$ we obtain the following:

$$
\begin{aligned}
&0\nu_1^* = 0\\
&0\nu_2^* = 0\\
&(5 - 2\nu_1^* - \nu_2^*)0 = 0\\
&(7 + \nu_1^* - 3\nu_2^*)5 = 0\\
&(10 - 5\nu_1^*)3 = 0
\end{aligned}
$$

Hence,

$$
\begin{aligned}
&\nu_1^* - 3\nu_2^* = -7\\
&5\nu_1^* = 10
\end{aligned}
$$

which yields $\boldsymbol{\nu}^* = (2, 3)$

□

6.6 Computer outputs and "what if" changes of single parameters

The goal of this section is to learn how to use AMPL to do sensitivity analysis. We will work with the following primal:

$$
\begin{aligned}
\min \quad & 5x_1 + 9x_2\\
\text{s.t.} \quad & x_1 + x_2 \geq 3\\
& x_1 - x_2 \leq 4\\
& x_1, x_2 \geq 0
\end{aligned}
$$

Our first task is to learn how to display information about the problem to develop some sensitivity analysis. We first need to add the sensitivity analysis features to cplex. We can do that by adding the following line after `option solver cplex;`:

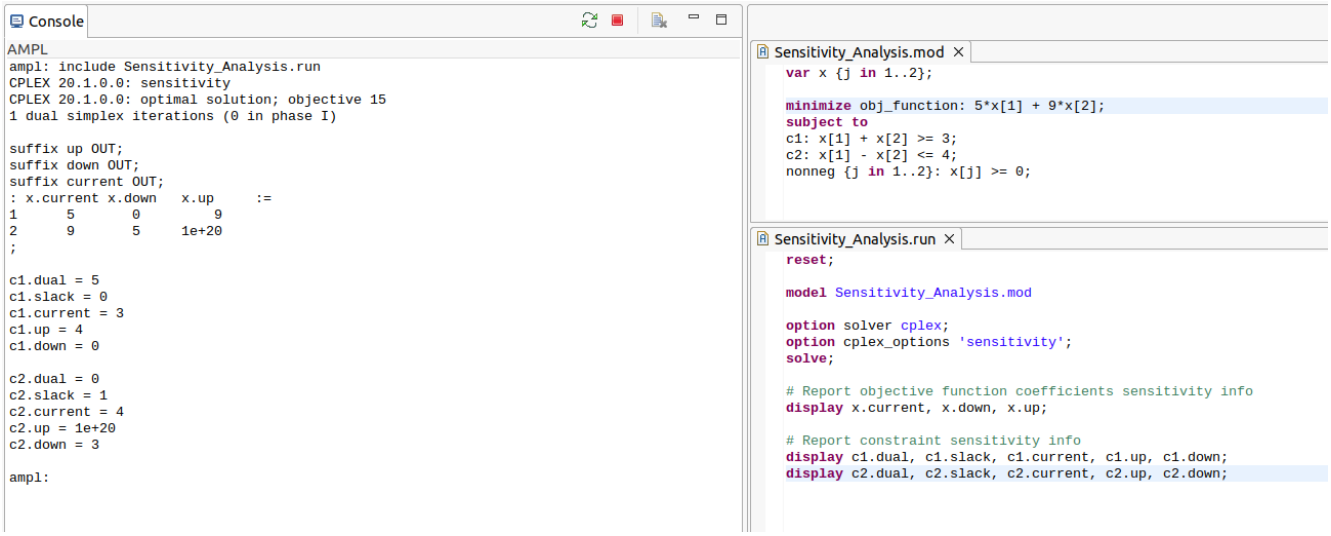
```
option cplex_options 'sensitivity';
```

To display additional information about the sensitivity to parameters, we add `.command` to the constraints/variables, as follows:

Part of the problem	Command	Description
Variables	`.current`	Objective function coefficient
	`.down`	Smallest value of the objective coefficient for which the current optimal basis remains optimal
	`.up`	Largest value of the objective coefficient for which the current optimal basis remains optimal
Constraints	`.current`	Right-hand side coefficient
	`.down`	Smallest value of the right-hand side coefficient for which the current optimal basis remains optimal

	`.up`	Largest value of the right-hand side coefficient for which the current optimal basis remains optimal
	`.dual`	Optimal value of the corresponding dual variable for the constraint
	`.slack`	Amount of slack in the primal constraint at optimality

We obtain the following output from AMPL:



From the output, we obtain the following information:

- The cost parameter of x_1 is $c_1 = 5$ and if we move it in the interval $[0, 9]$, the optimal solution will not change
- The cost parameter of x_2 is $c_2 = 9$ and if we move it in the interval $[5, 10^{20}]$, the optimal solution will not change
- The first constraint is active (slack $= 0$), its right-hand side is 3 and we may move it in the interval $[0, 4]$ without changing the optimal basis
- The second constraint is inactive (slack $= 1$), its right-hand side is 4 and we may move it in the interval $[3, 10^{20}]$ without changing the optimal basis
- The dual optimal solution is $\nu^* = (5, 0)$.

The ranges displayed by AMPL only say that if we change **one** constraint or variable, we may change it in the given ranges. If we move more than one we have to do a more in-deep study. If we want to perform too many changes, reoptimization might be the best solution.

6.7 Dual-simplex search

As the title insinuates, we will work on an improvement of the simplex algorithm using the dual. One reason for searching over the dual, is that the primal formulations often have more variables than constraints. Hence, the dual has a smaller number of variables than the primal and it might be considerably easier to solve.

Another reason is that finding an initial basis might be considerably easier for the dual problem. If we write our primal in standard form, then the dual has unrestricted variables. Therefore, just choosing a basis and inverting the corresponding matrix gives an initial basis. In other words, we do not need to check the nonnegativity constraints in basic solutions of the dual of a standard-form primal.

When we started studying simplex algorithm, we described it as an algorithm that searches for an optimal solution while maintaining feasibility. Similarly, we can describe the dual-simplex algorithms as the algorithms that maintain dual feasibility and complementary slackness while seeking primal feasibility.

We now develop the algorithm. Similarly to our approach to develop simplex, we go step by step and then we write the entire algorithm. We also start from a standard-form primal, which has the following dual:

$$\begin{array}{llll} (P) & \min & \boldsymbol{c}^T\boldsymbol{x} & \\ & \text{s.t} & A\boldsymbol{x} = \boldsymbol{b} & \\ & & \boldsymbol{x} \geq \boldsymbol{0} & \end{array} \qquad \begin{array}{llll} (D) & \max & \boldsymbol{b}^T\boldsymbol{\nu} & \\ & \text{s.t.} & A^T\boldsymbol{\nu} \leq \boldsymbol{c} & \\ & & \nu_i \text{ unrestricted} & \forall i \end{array}$$

For a given primal basis B, we may partition the primal and dual into basic and nonbasic primal solutions. Following the notation introduced in Section 5, we obtain the following primal and dual:

$$\begin{array}{llll} (P') & \min & \boldsymbol{c}_B^T\boldsymbol{x}_B + \boldsymbol{c}_N^T\boldsymbol{x}_N^T & \\ & \text{s.t} & B\boldsymbol{x}_B + N\boldsymbol{x}_N = \boldsymbol{b} & \\ & & \boldsymbol{x}_B \geq \boldsymbol{0} & \\ & & \boldsymbol{x}_N \geq \boldsymbol{0} & \end{array} \qquad \begin{array}{llll} (D') & \max & \boldsymbol{b}^T\boldsymbol{\nu} & \\ & \text{s.t.} & B^T\boldsymbol{\nu} \leq \boldsymbol{c}_B & \\ & & N^T\boldsymbol{\nu} \leq \boldsymbol{c}_N & \\ & & \nu_i \text{ unrestricted} & \forall i \end{array}$$

As mentioned above, in the dual-simplex algorithm we are interested in primal feasibility, dual feasibility and complementary slackness. In the following principle we write these conditions explicitly for the problems (P'), and (D').

Principle 6.14 (Principle 6.56 from the textbook)**.** *For the problems (P') and (D') presented above, we have the following expressions for primal/dual feasibility and complementary slackness:*

Primal feasibility:	***Dual feasibility:***	***Complementary slackness:***
$B\boldsymbol{x}_B + N\boldsymbol{x}_N = \boldsymbol{b}$	$B^T\boldsymbol{\nu} \leq \boldsymbol{c}_B$	$(\boldsymbol{c}_B - B^T\boldsymbol{\nu})^T\boldsymbol{x}_B = 0$
$\boldsymbol{x}_B \geq \boldsymbol{0}$	$N^T\boldsymbol{\nu} \leq \boldsymbol{c}_N$	$(\boldsymbol{c}_N - N^T\boldsymbol{\nu})^T\boldsymbol{x}_N = 0$
$\boldsymbol{x}_N \geq \boldsymbol{0}$		

Observe that we wrote the complementary slackness condition as a dot product, but each of the terms of the corresponding sum is nonnegative. Hence, the condition writen above is equivalent to the detailed complementary slackness conditions that we developed before.

As corollaries of the previous principle, we have the following facts.

- A solution $\boldsymbol{x} = (\boldsymbol{x}_B, \boldsymbol{x}_N)$ is a primal basic solution to the primal (P') if

  $$\boldsymbol{x}_N = \boldsymbol{0}, \qquad \text{and} \qquad \boldsymbol{x}_B = B^{-1}\boldsymbol{b}.$$

  The objective function value of a primal basic solution is $\boldsymbol{c}_B^T B^{-1}\boldsymbol{b}$, and the basic solution is primal feasible if $\boldsymbol{x}_B \geq \boldsymbol{0}$.

- To make sure that complementary slackness is always satisfied, we will work with

  $$\boldsymbol{\nu} = (B^{-1})^T\boldsymbol{c}_B$$

  This choice of $\boldsymbol{\nu}$ ensures that the first equation of complementary slackness is satisfied. The second one is trivially satisfied because $\boldsymbol{x}_N = \boldsymbol{0}$.

- The dual solution $\boldsymbol{\nu}$ defined above is dual feasible if it satisfies

  $$N^T\boldsymbol{\nu} \leq \boldsymbol{c}_N$$

- Whether or not $\boldsymbol{x} = (\boldsymbol{x}_B, \boldsymbol{x}_N)$ and $\boldsymbol{\nu}$ are primal and dual feasible, respectively, their objective function value is always the same. Indeed,

  $$\boldsymbol{c}^T\boldsymbol{x} = \boldsymbol{c}_B^T B^{-1}\boldsymbol{b} = \boldsymbol{\nu}^T\boldsymbol{b}$$

The principle and facts above represent the theoretical foundation of the dual simplex algorithm. Below we develop the steps in each iteration, that is, we determine a direction to move, a step size and an optimality condition.

To better understand the algorithm, let's work on an example as we build it.

Example 6.7. *Consider the following primal linear program in standard form:*

$$\begin{aligned} \min \quad & 2x_1 + 3x_2 \\ s.t. \quad & 3x_1 - 2x_2 - x_3 = 4 \\ & x_1 + 2x_2 - x_4 = 3 \\ & x_i \geq 0 \qquad \forall i \end{aligned}$$

and its dual

$$\begin{aligned} \max \quad & 4\nu_1 + 3\nu_2 \\ s.t. \quad & 3\nu_1 + \nu_2 \leq 2 \\ & -2\nu_1 + 2\nu_2 \leq 3 \\ & -\nu_1 \leq 0 \\ & -\nu_2 \leq 0 \\ & \nu_i \text{ unrestricted} \qquad \forall i \end{aligned}$$

Then, the matrix A and the vectors $\boldsymbol{c}$ and $\boldsymbol{b}$ are:

$$A = \begin{bmatrix} 3 & -2 & -1 & 0 \\ 1 & 2 & 0 & -1 \end{bmatrix} \qquad \boldsymbol{b} = \begin{pmatrix} 4 \\ 3 \end{pmatrix} \qquad \boldsymbol{c} = \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix}$$

In the example, observe that the dual constraints corresponding to the primal variables x_3 and x_4 (that could be slack variables of $\geq$ constraints) are sign constraints on ν_1 and ν_2. Then, we obtain a dual consistent with the dual we would obtain from a primal which is not in standard form.

Choosing an improving direction

Recall that we wish to maintain dual feasibility and complementary slackness while seeking primal feasibility. Then, the direction to move will be represented by a primal variable r which is infeasible, that is, that satisfies $x_r < 0$.

Principle 6.15 (Principle 6.62 from the textbook)**.** *Dual simplex search adopts the direction $\Delta\boldsymbol{\nu} = -\boldsymbol{r}$ for a minimizing primal, where $\boldsymbol{r}$ is the row of B^{-1} corresponding to an infeasible primal solution $x_r < 0$.*

Let's exemplify the computation of the dual direction with Example 6.7. If we choose the basis $\{x_3, x_4\}$ we obtain the matrix

$$B = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} = B^{-1}$$

and the primal basic solution

$$\boldsymbol{x}_B = B^{-1}\boldsymbol{b} = \begin{pmatrix} -4 \\ -3 \end{pmatrix}$$

which is clearly primal infeasible because both elements are negative.

If we choose the infeasible component $r = 3$, then the dual direction is the first row (because the basis is $\{x_3, x_4\}$ and x_3 is the first variable) of B^{-1} with a negative sign, that is,

$$\Delta\boldsymbol{\nu} = -\boldsymbol{r} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Before moving on with the method, let's make sure that the direction $\Delta\boldsymbol{\nu}$ that we chose is an improving direction. In other words, let's make sure that the direction makes the dual solution better (bigger, since it is a maximization).

We know that the r^{th} primal basic solution satisfies $x_r = \boldsymbol{r}^T\boldsymbol{b} < 0$ by definition of matrix multiplication. Then, if we move in the direction $\Delta\boldsymbol{\nu}$ with a step-size $\lambda \geq 0$, we obtain the following new dual objective value:

$$\begin{aligned}\boldsymbol{b}^T\boldsymbol{\nu}^{new} &= \boldsymbol{b}^T\boldsymbol{\nu}^{old} + \lambda\boldsymbol{b}^T\Delta\boldsymbol{\nu} \\ &= \boldsymbol{b}^T\boldsymbol{\nu}^{old} - \lambda\boldsymbol{b}^T\boldsymbol{r} \\ &= \boldsymbol{b}^T\boldsymbol{\nu}^{old} - \lambda x_r\end{aligned}$$

The dual objective changes $-x_r$, which is a positive quantity because $x_r < 0$. Then, $\boldsymbol{b}^T\boldsymbol{\nu}^{new} \geq \boldsymbol{b}^T\boldsymbol{\nu}^{old}$, that is, the dual objective function increases in the direction $\Delta\boldsymbol{\nu}$. Since the dual is a maximization problem, the dual objective is improving.

Determining the step size

In simplex algorithm, we chose the step size to maintain primal feasibility. In this case, we choose the step size to maintain dual feasibility. Since $\boldsymbol{\nu} = B^{-1}\boldsymbol{c}_B$, we only need to make sure that

$$\begin{aligned}& N^T(\boldsymbol{\nu} + \lambda\Delta\boldsymbol{\nu}) \leq \boldsymbol{c}_N \\ \iff\quad & \lambda N^T\Delta\boldsymbol{\nu} \leq \boldsymbol{c}_N - N^T\boldsymbol{\nu} \\ \iff\quad & \lambda \underbrace{(\boldsymbol{A}^{(j)})^T\Delta\boldsymbol{\nu}}_{\text{change in reduced cost } \overline{c}_j} \leq \underbrace{c_j - (\boldsymbol{A}^{(j)})^T\boldsymbol{\nu}}_{\text{reduced cost } \overline{c}_j} \qquad \text{for all nonbasic } j \\ \iff\quad & \lambda\Delta\overline{c}_j \leq \overline{c}_j \qquad \text{for all nonbasic } j\end{aligned}$$

To solve for λ we split in two cases: when $\Delta\overline{c}_j$ is positive and negative. If $\Delta\overline{c}_j > 0$, then we obtain

$$\lambda \geq \frac{\overline{c}_j}{\Delta\overline{c}_j},$$

which is meaningless because the right-hand side of the inequality is negative and we are seeking $\lambda > 0$. In particular, if $\Delta\overline{c}_j > 0$ for all nonbasic j, then the algorithm would detect unboundedness because there are no restrictions on λ.

If $\Delta\overline{c}_j < 0$, we obtain

$$\lambda \leq \frac{\overline{c}_j}{\Delta\overline{c}_j},$$

and these inequalities are satisfied if we choose the smallest right-hand side among nonbasic variables j with $\Delta\overline{c}_j < 0$. Therefore, we have the following principle.

Principle 6.16 (Principle 6.63 from the textbook)**.** *A step $\lambda > 0$ in direction $\Delta\boldsymbol{\nu}$ will change the current primal reduced cost by $\Delta\overline{\boldsymbol{c}} = -N^T\Delta\boldsymbol{\nu}$. Then, λ must satisfy:*

$$\lambda \leftarrow \frac{\overline{c}_p}{-\Delta\overline{c}_p} = \min\left\{\frac{\overline{c}_j}{-\Delta\overline{c}_j} : \ \Delta\overline{c}_j < 0, \ j \text{ nonbasic}\right\}$$

If no limit is encountered, the dual is unbounded, which implies that the primal is infeasible.

In the definition of $\overline{\boldsymbol{c}}$, observe that the vector $\boldsymbol{\nu}$ plays the same role as the pricing vector that we defined in Section 5.6, when we discussed revised simplex.

Following with our example, we have

$$\Delta\overline{\boldsymbol{c}} = -N^T\Delta\boldsymbol{\nu} = \begin{bmatrix}-3 & -1 \\ 2 & -2\end{bmatrix}\begin{pmatrix}1 \\ 0\end{pmatrix} = \begin{pmatrix}-3 \\ 2\end{pmatrix}$$

We additionally need to compute the reduced costs, which we compute as follows:

$$\overline{\boldsymbol{c}} = \boldsymbol{c}_N - N^T(B^{-1})\boldsymbol{c}_B = \boldsymbol{c}_N = \begin{pmatrix}2 \\ 3\end{pmatrix}$$

Then, the step size is

$$\lambda = \frac{\bar{c}_1}{-\Delta\bar{c}_1} = \frac{2}{-(-3)} = \frac{2}{3}$$

With this update, we increase the value of the infeasible variable $x_r < 0$ to 0.

In the development of the dual simplex algorithm we used a minimizing primal. If we had a maximizing primal with standard form

$$\begin{aligned} (P_2) \quad \max \quad & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.} \quad & A\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

then its dual is

$$\begin{aligned} (D_2) \quad \min \quad & \boldsymbol{b}^T\boldsymbol{\nu} \\ \text{s.t.} \quad & A^T\boldsymbol{\nu} \geq \boldsymbol{c} \\ & \nu_i \text{ unrestricted} \qquad \forall i \end{aligned}$$

Therefore, for a given basis B the problems become

$$\begin{aligned} (P_2') \quad \max \quad & \boldsymbol{c}_B^T\boldsymbol{x}_B + \boldsymbol{c}_N^T\boldsymbol{x}_N^T \\ \text{s.t} \quad & B\boldsymbol{x}_B + N\boldsymbol{x}_N = \boldsymbol{b} \\ & \boldsymbol{x}_B \geq \boldsymbol{0} \\ & \boldsymbol{x}_N \geq \boldsymbol{0} \end{aligned} \qquad \begin{aligned} (D_2') \quad \min \quad & \boldsymbol{b}^T\boldsymbol{\nu} \\ \text{s.t.} \quad & B^T\boldsymbol{\nu} \geq \boldsymbol{c}_B \\ & N^T\boldsymbol{\nu} \geq \boldsymbol{c}_N \\ & \nu_i \text{ unrestricted} \qquad \forall i \end{aligned}$$

Observe that the only differences between the dual of the minimizing primal (D') and the dual of the maximizing primal (D_2') is the objective and the orientation of the inequalities in the main constraints ($\leq$ vs $\geq$). Then, one can follow the same intuitions to construct a dual-simplex method. We skip the details for brevity.

Below we show the steps of the dual-simplex method, and we point out the differences between a minimizing and maximizing primal.

Algorithm 6.1 (Dual simplex search)**.** *For a primal in standard form, follow these steps:*

0. ***Initialization:***
   - 0.1 *Initialize solution index* $t \leftarrow 0$
   - 0.2 *Choose a dual-feasible basis, compute the matrices* B, N *and* B^{-1}
   - 0.3 *Set nonbasic primal variables* $\boldsymbol{x}_N^{(0)} \leftarrow \boldsymbol{0}$ *and basic primal variables* $\boldsymbol{x}_B^{(0)} \leftarrow B^{-1}\boldsymbol{b}$
   - 0.4 *Compute dual basic solution* $\boldsymbol{\nu}^{(0)} \leftarrow (B^{-1})^T\boldsymbol{c}_B$
1. ***Optimality:***
   - *If* $\boldsymbol{x}_B^{(t)} \geq \boldsymbol{0}$*, STOP. Current solution is optimal. The primal optimal solution is* $\boldsymbol{x}^{(t)}$*, the dual optimal solution is* $\boldsymbol{\nu}^{(t)}$ *and the optimal value is* $\boldsymbol{c}^T\boldsymbol{x}^{(t)} = \boldsymbol{b}^T\boldsymbol{\nu}^{(t)}$*.*
   - *If not, choose a primal infeasible basic variable* r *that satisfies* $x_r < 0$*.*
2. ***Dual-simplex direction:*** *Let* $\boldsymbol{r}$ *be the row of* B^{-1} *corresponding to the variable* r *and construct the dual-simplex direction:* $\Delta\boldsymbol{\nu} \leftarrow +\boldsymbol{r}$ *for a maximizing primal and* $\Delta\boldsymbol{\nu} \leftarrow -\boldsymbol{r}$ *for a minimizing primal.*

   Also, compute the change to the nonbasic reduced costs: $\Delta\bar{\boldsymbol{c}}_N \leftarrow -N^T\Delta\boldsymbol{\nu}$*.*
3. ***Step size:***

- *If* $\Delta\bar{\boldsymbol{c}}_N \leq \mathbf{0}$ *for a maximizing primal/*$\Delta\bar{\boldsymbol{c}}_N \geq \mathbf{0}$ *for a minimizing primal, STOP. Dual is unbounded and, hence, primal is infeasible*
- *Otherwise, choose a step size* λ *as follows:*
  - *For a maximizing primal*

$$\lambda \leftarrow \frac{-\bar{c}_p}{\Delta\bar{c}_p} = \min\left\{\frac{-\bar{c}_j}{\Delta\bar{c}_j} : \ \Delta\bar{c}_j > 0, \ j \ \textit{nonbasic}\right\}$$

  - *For a minimizing primal*

$$\lambda \leftarrow \frac{-\bar{c}_p}{\Delta\bar{c}_p} = \min\left\{\frac{-\bar{c}_j}{\Delta\bar{c}_j} : \ \Delta\bar{c}_j < 0, \ j \ \textit{nonbasic}\right\}$$

4. ***New solution and basis:***

4.1 Replace x_r *by* x_p *in the basis, update* B, N *and* B^{-1} *and compute the new basic primal solution* $\boldsymbol{x}^{(t+1)}$.

4.2 The corresponding dual basic solution is $\boldsymbol{\nu}^{(t+1)} \leftarrow \boldsymbol{\nu}^{(t)} + \lambda\Delta\boldsymbol{\nu}$.

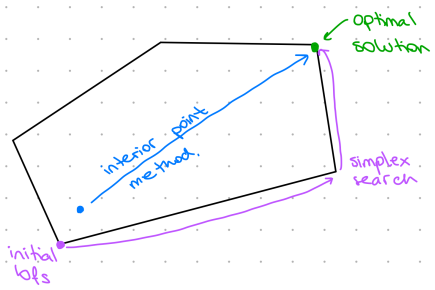
4.3 Advance $t \leftarrow t + 1$.

7 Interior Point Methods for Linear Programming

[This section is based on Chapter 7 of the textbook]

We have spent a lot of energy studying the simplex algorithm, and its variant, the dual-simplex algorithm. This algorithm exploits the idea that there will always be an optimal solution at an extreme point of a linear program. In this chapter we will learn a different approach: we will search through the interior of the feasible region.

One motivation to do such thing, is that the number of extreme points of a polyhedron is exponential in the number of variables. Then, in the worst case, simplex algorithm searches through a very large number of basic feasible solutions. Another motivation is that when we search on the extreme points, we are taking a long route around the boundary of the feasible region. If we go through the interior, instead, we can go directly to an optimal extreme point. For example, see the figure below.



Let's start building some intuition about the search for an optimal solution through the interior.

7.1 Searching through the interior

As usual in this course, we will use an example to construct the main ideas of this section.

Example 7.1 (Application 7.1 from the textbook)**.** *Each year, Frannie sells up to 3 cords of firewood from her small woods. One potential customer has offered her \$90 per half-cord, and another \$150 per full cord. Our concern is how much Frannie should sell to each customer to maximize income, assuming that each will buy as much as he or she can.*

Let's first model Frannie's problem. Define the following decision variables:

$$\begin{array}{lcl} x_1 & = & \textit{Number of half-cords sold to customer 1} \\ x_2 & = & \textit{Number of cords sold to customer 2} \end{array}$$

Then, we obtain the following linear program:

$$\begin{array}{ll} \max & 90x_1 + 150x_2 \\ s.t. & \frac{1}{2}x_1 + x_2 \leq 3 \\ & x_1, x_2 \geq 0 \end{array}$$

When we search through the interior of a feasible region, we are only evaluating points that are away from the boundary. Hence, any direction we take is feasible in a small neighborhood. This means that we can move in the direction of the most rapid improvement, that is, the gradient of the objective function for maximizing problems, and the negative gradient for minimizing problems.

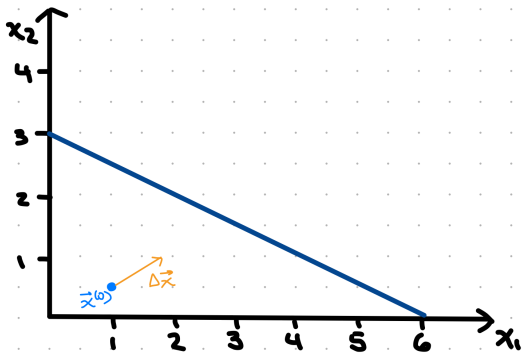
For example, in Frannie's problem, an interior point is

$$\boldsymbol{x}^{(0)} = \left(1, \frac{1}{2}\right)$$

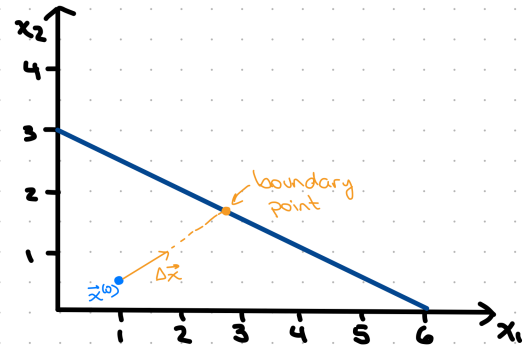
and the direction of most rapid improvement is

$$\Delta \boldsymbol{x} = \boldsymbol{c} = (90, 150)$$

The point $\boldsymbol{x}^{(0)}$ and the direction $\Delta \boldsymbol{x}$ are shown in the following picture.



Now, if we take the largest possible step in the direction $\Delta \boldsymbol{x}$ (which is what we used to do in simplex), we will end up at the boundary and will not reach an optimal solution, as shown in the next picture.



Then, in the next step we would not have the advantage that any direction is feasible. Hence, we do not want to give such a step in interior point methods. Instead, we wish to stay in the interior for a while, until we reach a point closer to the optimal solution.

Checking if a point is in the interior of a large optimization problem is a difficult task in general. Then, we will write our problem in standard form. Recall that an LP in standard form is written as:

$$\begin{aligned} \min / \max \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} \quad & A\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

Then, a point $\overline{\boldsymbol{x}}$ is in the interior of the feasible region if and only if $\overline{\boldsymbol{x}} > \boldsymbol{0}$, that is, if it satisfies all the inequality constraints strictly.

For example, Frannie's problem in standard form is

$$\begin{aligned} \max \quad & 90x_1 + 150x_2 \\ \text{s.t.} \quad & \frac{1}{2}x_1 + x_2 + x_3 = 3 \\ & x_i \geq 0 \qquad \forall i \end{aligned}$$

and $\boldsymbol{x}^{(0)} = (1, 1/2, 2)$ is an interior point.

Unfortunately, there is no free lunch. Writing the problem in standard form helps us to decide whether a feasible point is in the interior or the boundary of the feasible region. However, choosing directions that are both, feasible and improving, gets a bit trickier because there are more equality constraints to preserve.

When we studied simplex method, we learned that a direction $\Delta \boldsymbol{x}$ that satisfies

$$A\Delta \boldsymbol{x} = \boldsymbol{0}$$

preserves the equality constraints. That is, if $\boldsymbol{x}$ satisfies $A\boldsymbol{x} = \boldsymbol{b}$, then

$$A(\boldsymbol{x} + \Delta \boldsymbol{x}) = \boldsymbol{b}$$

as well.

Now, the question is: how to compute a direction that preserves the equality constraints **and** is very close to the gradient direction $\boldsymbol{c}$?

Principle 7.1 (Principles 7.5-7.6 from the textbook)**.** *The closest direction to a vector $\boldsymbol{d}$ that satisfies the equality constraints $A\Delta \boldsymbol{x} = \boldsymbol{0}$ is the projection of $\boldsymbol{d}$ on the space generated by the equality constraints. Such projection can be computed as*

$$\Delta \boldsymbol{x} = P\boldsymbol{d}, \qquad \textit{where} \quad P = I - A^T(AA^T)^{-1}A$$

P is called projection matrix.

In Frannie's problem, we are interested in computing the projection of the standard form gradient $\boldsymbol{c} = (90, 150, 0)$ on the system of equality constraints $A\Delta \boldsymbol{x} = \boldsymbol{0}$, with

$$A = \begin{bmatrix} \frac{1}{2} & 1 & 1 \end{bmatrix}$$

Let's first compute the projection matrix P. We start computing AA^T and its inverse. We obtain

$$AA^T = \begin{bmatrix} \frac{1}{2} & 1 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{2} \\ 1 \\ 1 \end{bmatrix} = \frac{9}{4} \quad \Longrightarrow \quad (AA^T)^{-1} = \frac{4}{9}$$

Then,

$$\begin{aligned} P &= I - A^T(AA^T)^{-1}A \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \frac{4}{9} \begin{bmatrix} \frac{1}{2} \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} \frac{1}{2} & 1 & 1 \end{bmatrix} \end{aligned}$$

$$= \frac{1}{9}\begin{bmatrix} 8 & -2 & -2 \\ -2 & 5 & -4 \\ -2 & 4 & 4 \end{bmatrix}$$

Therefore, the projection of the vector $\boldsymbol{c}$ on the equality constraints is

$$\Delta \boldsymbol{x} = P\boldsymbol{c} = \frac{1}{9}\begin{bmatrix} 8 & -2 & -2 \\ -2 & 5 & -4 \\ -2 & -4 & 4 \end{bmatrix}\begin{pmatrix} 90 \\ 150 \\ 0 \end{pmatrix} = \frac{1}{9}\begin{pmatrix} 420 \\ 570 \\ -780 \end{pmatrix}$$

Since the objective function of linear problems is constant, we know that the projection of $\boldsymbol{c}$ on the equality constraints is always an improving feasible direction.

7.2 Scaling with the current solution

In the previous section we learned how to compute a good direction to move, that is, an improving direction that will maintain feasibility. The next question we need to answer is how much to move into that direction.

When we were doing simplex, we wanted to go as far as possible in order to maintain feasibility and reach another extreme point. In this case, we want to remain in the interior of the feasible region until we are close to optimality. Hence, we need to know how far away the current solution is from the boundary of every inequality constraint. We will do that using affine scaling, as follows.

Definition 7.1 (Definition 7.8 from the textbook)**.** *At the current solution $\boldsymbol{x}^{(t)} > \boldsymbol{0}$, affine scaling transforms points $\boldsymbol{x}$ into $\boldsymbol{y}$ defined by*

$$y_j \triangleq \frac{x_j}{x_j^{(t)}} \qquad \forall j$$

or, equivalently,

$$\boldsymbol{y} \triangleq (X^{(t)})^{-1}\boldsymbol{x},$$

where $X^{(t)}$ denotes a square matrix with the components of $\boldsymbol{x}^{(t)}$ on its diagonal.

Then, the current solution is equally distant from every inequality constraint $\boldsymbol{x} \geq \boldsymbol{0}$.

In Frannie's example, we had $\boldsymbol{x}^{(0)} = (1, 1/2, 2)$. Then, after applying affine scaling we obtain:

$$\boldsymbol{y} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 2 \end{bmatrix}^{-1}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & \frac{1}{2} \end{bmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_1 \\ 2x_2 \\ \frac{x_3}{2} \end{pmatrix}$$

Hence, the scaled version of $\boldsymbol{x}^{(0)}$ is $\boldsymbol{y}^{(0)} = (1, 1, 1)$, which is at equivalent distance from the boundary of the constraints $\boldsymbol{y} \geq \boldsymbol{0}$.

When we apply affine scaling, we are reshaping the feasible region to gain tractability. Then, the objective function and the constraints will change, as specified in the following definition.

Definition 7.2 (Definition 7.10 from the textbook)**.** *At the current feasible solution $\boldsymbol{x}^{(t)} > \boldsymbol{0}$, the affine-scaled version of a standard-form linear program is*

$$\begin{aligned} \min / \max \quad & (\boldsymbol{c}^{(t)})^T\boldsymbol{y} \\ s.t. \quad & A^{(t)}\boldsymbol{y} = \boldsymbol{b} \\ & \boldsymbol{y} \geq \boldsymbol{0} \end{aligned}$$

where

$$(\boldsymbol{c}^{(t)})^T \triangleq \boldsymbol{c}^T X^{(t)}, \qquad A^{(t)} \triangleq AX^{(t)}$$

Observe that the right-hand side vector $\boldsymbol{b}$ remains unchanged.

As an example, let's write the affine-scaled form of Frannie's problem corresponding to $\boldsymbol{x}^{(t)} = (3, 1/2, 1)$. We have:

$$X^{(t)} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Then,

$$(\boldsymbol{c}^{(t)})^T = \begin{pmatrix} 90 & 150 & 0 \end{pmatrix}\begin{bmatrix} 3 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{pmatrix} 270 & 75 & 0 \end{pmatrix}$$

$$A^{(t)} = \begin{bmatrix} \frac{1}{2} & 1 & 1 \end{bmatrix}\begin{bmatrix} 3 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{3}{2} & \frac{1}{2} & 1 \end{bmatrix}$$

Hence, Frannie's problem scaled affinely at $\boldsymbol{x}^{(t)} = (3, 1/2, 1)$ is

$$\begin{aligned} \max \quad & 270y_1 + 75y_2 \\ \text{s.t.} \quad & \frac{3}{2}y_1 + \frac{1}{2}y_2 + y_3 = 3 \\ & y_1, y_2, y_3 \geq 0 \end{aligned}$$

We saw before that it is essential to project the moving direction on the equality constraints. If we scale the problem, we also must scale the projection. In the following principle we show how.

Principle 7.2 (Principle 7.11 from the textbook)**.** *The projection of a direction $\boldsymbol{d}$ onto affine-scaled conditions $A^{(t)}\Delta\boldsymbol{y} = \boldsymbol{0}$ preserving the linear equalities $A^{(t)}\boldsymbol{y} = \boldsymbol{b}$ can be computed as*

$$\Delta\boldsymbol{y} = P^{(t)}\boldsymbol{d}, \qquad \textit{where} \quad P^{(t)} = I - (A^{(t)})^T(A^{(t)}(A^{(t)})^T)^{-1}A^{(t)}$$

7.3 Affine scaling search algorithm

In the previous section we sketched how to choose good directions to move in interior point methods. To complete the algorithm, all we need to do is decide the step size in every iteration, and determine a termination condition.

To determine the step size, recall that we want to move in an improving direction while staying in the interior of the feasible region. An easy way to make sure we stay in the interior, is to use the scaled vector $\boldsymbol{y}$. Observe that, by definition, if we scale at $\boldsymbol{x}^{(t)}$, then $\boldsymbol{y}^{(t)} = \mathbf{1}$. Hence, we know that the vector $\boldsymbol{y}^{(t)}$ is at distance 1 from the boundary of all the inequality constraints $\boldsymbol{y} \geq \mathbf{0}$ and it makes sense to use $\Delta\boldsymbol{y}$ to determine the step size. Indeed, the affine scaling search algorithm uses step size

$$\lambda = \frac{1}{\|\Delta \boldsymbol{y}\|} = \frac{1}{\|(X^{(t)})^{-1}\Delta \boldsymbol{x}\|}$$

where $\|\boldsymbol{d}\|$ denotes the Euclidean norm of the vector $\boldsymbol{d}$, that is,

$$\|\boldsymbol{d}\| = \sqrt{\sum_{i=1}^{n} d_i^2}$$

Regarding the termination condition, if we get lucky, we can end up exactly at an optimal solution after an update. However, this is very rare. More frequently, we will realize that the new solutions are very similar to the previous solutions. Hence, the algorithm must be near an optimal solution and we stop. Therefore, we obtain the following algorithm:

Algorithm 7.1 (Affine scaling search for linear programs)**.** *Starting from a linear optimization problem in standard form,*

0. ***Initialization:***
   Set $t \leftarrow 0$ and choose an initial interior point $\boldsymbol{x}^{(0)}$.
1. ***Optimality:***
   - *If any component of $\boldsymbol{x}^{(t)}$ is 0, then STOP. Current solution is optimal.*
   - *If recent steps have not made significant change in the solution value, then STOP. Current point $\boldsymbol{x}^{(t)}$ is very nearly optimal*
   - *Else, move to next step.*
2. ***Move direction:***
   - *In a minimization problem, let $\Delta\boldsymbol{x}^{(t+1)} = -X^{(t)}P^{(t)}\boldsymbol{c}^{(t)}$*
   - *In a maximization problem, let $\Delta\boldsymbol{x}^{(t+1)} = +X^{(t)}P^{(t)}\boldsymbol{c}^{(t)}$*
3. ***Step size:***
   - *If $\Delta\boldsymbol{x}^{(t+1)} \geq \mathbf{0}$, then STOP. Model is unbounded.*
   - *Otherwise, compute step size $\lambda \leftarrow \dfrac{1}{\|(X^{(t)})^{-1}\Delta\boldsymbol{x}^{(t+1)}\|}$*
4. ***Move to a new solution:***
   Compute the new solution $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} + \lambda\Delta\boldsymbol{x}^{(t+1)}$, update $t \leftarrow t+1$ and go to step 1.

Before moving on to the next algorithm, let's do an example.

Example 7.2 (based on Examples 7.8-7.10 from the textbook)**.** *After 7 iterations in an improving search of the linear program*

$$\begin{aligned} \min \quad & -3x_1 + 9x_3 \\ s.t. \quad & -x_1 + x_3 = 3 \\ & x_1 + 2x_2 = 4 \\ & x_i \geq 0 \qquad \forall i \end{aligned}$$

we reach $\boldsymbol{x}^{(7)} = (2, 1, 5)$. Run one iteration of the affine scaling search.

Solution. Our current point $\boldsymbol{x}^{(7)}$ does not satisfy the optimality conditions, so we compute a moving direction. Since this is a minimization problem, we use $\Delta\boldsymbol{x}^{(t+1)} = -X^{(t)}P^{(t)}\boldsymbol{c}^{(t)}$. We first compute each of the terms on the right-hand side.

We have

$$X^{(7)} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

Then,

$$\boldsymbol{c}^{(7)} = X^{(7)}\boldsymbol{c} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 5 \end{bmatrix} \begin{pmatrix} -3 \\ 0 \\ 9 \end{pmatrix} = \begin{pmatrix} -6 \\ 0 \\ 45 \end{pmatrix}$$

$$A^{(7)} = AX^{(7)} = \begin{bmatrix} -1 & 0 & 1 \\ 1 & 2 & 0 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 5 \end{bmatrix} = \begin{bmatrix} -2 & 0 & 5 \\ 2 & 2 & 0 \end{bmatrix}$$

Now we compute the projection matrix $P^{(7)}$. We have

$$A^{(7)}(A^{(7)})^T = \begin{bmatrix} -2 & 0 & 5 \\ 2 & 2 & 0 \end{bmatrix} \begin{bmatrix} -2 & 2 \\ 0 & 2 \\ 5 & 0 \end{bmatrix} = \begin{bmatrix} 29 & -4 \\ -4 & 8 \end{bmatrix}$$

$$\implies \left(A^{(7)}(A^{(7)})^T\right)^{-1} = \frac{1}{216}\begin{bmatrix} 8 & 4 \\ 4 & 29 \end{bmatrix}$$

$$\implies P^{(7)} = I - (A^{(7)})^T \left(A^{(7)}(A^{(7)})^T\right)^{-1} A^{(7)} = \frac{1}{54}\begin{bmatrix} 25 & -25 & 10 \\ -25 & 25 & -10 \\ 10 & -10 & 4 \end{bmatrix}$$

Therefore, since this is a minimization problem, we obtain

$$\Delta \boldsymbol{x}^{(8)} = -X^{(7)}P^{(7)}\boldsymbol{c}^{(7)} = -\frac{1}{54}\begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 5 \end{bmatrix}\begin{bmatrix} 25 & -25 & 10 \\ -25 & 25 & -10 \\ 10 & -10 & 4 \end{bmatrix}\begin{pmatrix} -6 \\ 0 \\ 45 \end{pmatrix} = \frac{1}{54}\begin{pmatrix} -600 \\ 300 \\ -600 \end{pmatrix} = \frac{1}{9}\begin{pmatrix} -100 \\ 50 \\ -100 \end{pmatrix}$$

Since the first and third element of our direction are negative numbers, we do not detect unboundedness in this step. Then, we compute the step size. We first compute

$$(X^{(7)})^{-1}\Delta \boldsymbol{x}^{(8)} = \frac{1}{9}\begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{5} \end{bmatrix}\begin{pmatrix} -100 \\ 50 \\ -100 \end{pmatrix} = \frac{1}{9}\begin{pmatrix} -50 \\ 50 \\ -20 \end{pmatrix}$$

$$\implies \left\|(X^{(7)})^{-1}\Delta \boldsymbol{x}^{(8)}\right\| = \sqrt{\frac{1}{81}\left(50^2 + 50^2 + 20^2\right)} = \sqrt{\frac{5400}{81}}$$

$$\implies \lambda = \frac{1}{\left\|(X^{(7)})^{-1}\Delta \boldsymbol{x}^{(8)}\right\|} = \sqrt{\frac{81}{5400}} \approx 0.1225$$

Therefore, the new point is

$$\boldsymbol{x}^{(8)} = \boldsymbol{x}^{(7)} + \lambda \Delta \boldsymbol{x}^{(8)} = \begin{pmatrix} 2 \\ 1 \\ 5 \end{pmatrix} + \sqrt{\frac{81}{5400}}\frac{1}{9}\begin{pmatrix} -50 \\ 50 \\ -20 \end{pmatrix} \approx \begin{pmatrix} 1.32 \\ 1.68 \\ 4.73 \end{pmatrix}$$

□

7.4 Log barrier methods for interior point search

The above is just one way to search optimal solutions through the interior of the feasible region of a linear program. We now learn another way. The whole idea is to remain far away from the boundary of the feasible region, that is, we want values of x_j that are larger than 0. In affine scaling, we did that dividing by the values of x_j in each iteration. Here we create another auxiliary problem, using properties of the logarithm to penalize solutions that are close to the boundary. Specifically, we formulate the log barrier problem as follows.

Principle 7.3 (Principle 7.17 from the textbook)**.** *In a standard form maximization problem with objective function $\boldsymbol{c}^T\boldsymbol{x}$, we modify the objective function with a log barrier as follows:*

$$\max \quad \boldsymbol{c}^T\boldsymbol{x} + \mu \sum_j \log(x_j)$$

where $\mu > 0$ is a specified constant. Similarly, we modify the objective function of a minimization problem as follows:

$$\min \quad \boldsymbol{c}^T\boldsymbol{x} - \mu \sum_j \log(x_j)$$

Let's do an example and compare the optimal value for two different solutions.

Example 7.3. *Let's consider Frannie's problem. Recall that the standard form is*

$$\begin{aligned} \max \quad & 90x_1 + 150x_2 \\ s.t. \quad & \frac{1}{2}x_1 + x_2 + x_3 = 3 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Construct the associated log-barrier problem and compare the objective value obtained with $\mu = 64$ at $\boldsymbol{x}^{(0)} = (2, 1, 1)$ and $\boldsymbol{x}^{(1)} = (0.01, 2.99, 0.005)$. Repeat the exercise for $\mu = 10$.

Solution. We first construct the log barrier problem. Since it is a maximization problem, we add the barrier, as follows:

$$\begin{aligned} \max \quad & 90x_1 + 150x_2 + \mu\left(\log(x_1) + \log(x_2) + \log(x_3)\right) \\ \text{s.t.} \quad & \frac{1}{2}x_1 + x_2 + x_3 = 3 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Let

$$f(\boldsymbol{x}, \mu) = 90x_1 + 150x_2 + \mu\left(\log(x_1) + \log(x_2) + \log(x_3)\right).$$

For $\boldsymbol{x}^{(0)}$, the true objective value and the barrier objective value are:

$$f(\boldsymbol{x}^{(0)}, 0) = \boldsymbol{c}^T\boldsymbol{x}^{(0)} = 330$$
$$f(\boldsymbol{x}^{(0)}, 64) = 374.36$$
$$f(\boldsymbol{x}^{(0)}, 10) = 336.93$$

And for $\boldsymbol{x}^{(1)}$, we obtain

$$f(\boldsymbol{x}^{(1)}, 0) = \boldsymbol{c}^T\boldsymbol{x}^{(1)} = 449.4$$
$$f(\boldsymbol{x}^{(1)}, 64) = -114.33$$
$$f(\boldsymbol{x}^{(1)}, 10) = 361.32$$

Observe that $\boldsymbol{x}^{(1)}$ is near the boundary of the feasible region because $x_1^{(1)}$ and $x_3^{(1)}$ are close to zero.

□

Observe that, the larger the value of μ, the larger the penalty for using values of $\boldsymbol{x}$ that are near the boundary. However, if we start with a small value of μ, it means we may search through points that are very close to the boundary. Then, an ideal algorithm would start with a large μ and, as we get closer to the optimal solution, we should decrease μ.

Also observe that, by adding the barrier term, the problem became nonlinear. Then, we need nonlinear programming tools to solve it. We won't derive all the steps, but the basic idea is that we need to move in directions that make sense for second-order approximation of the objective function, that is, we use the first and second derivative. We obtain the following algorithm.

Algorithm 7.2 (Newton step for log barrier search for linear programs)**.** *Starting from a linear optimization problem in standard form,*

0. ***Initialization:***
   Set $t \leftarrow 0$, choose an initial interior point $\boldsymbol{x}^{(0)}$ and a relatively large barrier multiplier μ.
1. ***Move direction:***
   - *In a minimization problem, set*

$$\Delta\boldsymbol{x}^{(t+1)} \leftarrow -X^{(t)}P^{(t)}\begin{pmatrix} c_1^{(t)} - \mu \\ c_2^{(t)} - \mu \\ \vdots \\ c_n^{(t)} - \mu \end{pmatrix}$$

   - *In a maximization problem, set*

$$\Delta\boldsymbol{x}^{(t+1)} \leftarrow +X^{(t)}P^{(t)}\begin{pmatrix} c_1^{(t)} + \mu \\ c_2^{(t)} + \mu \\ \vdots \\ c_n^{(t)} + \mu \end{pmatrix}$$

2. ***Step size:***
   Choose

$$\lambda \leftarrow \min\left\{\frac{1}{\mu},\, 0.9\lambda_{\max}\right\}$$

   where

$$\lambda_{\max} \leftarrow \min\left\{\frac{x_j^{(t)}}{-\Delta x_j^{(t+1)}} :\ \Delta x_j^{(t+1)} < 0\right\}$$

3. ***Compute new solution:***
   Compute $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} + \lambda\Delta\boldsymbol{x}^{(t+1)}$
4. ***Inner loop:***
   - *If $\boldsymbol{x}^{(t+1)}$ is distant from $\boldsymbol{x}^{(t)}$ (that is, if $\boldsymbol{x}^{(t+1)}$ is far from the optimal solution), increment $t \leftarrow t+1$ and go to step 1.*

- *Else, that is, if $\boldsymbol{x}^{(t+1)}$ is close to $\boldsymbol{x}^{(t)}$, go to step 5.*

5. ***Outer loop:***
   - *If barrier multiplier μ is close to 0, STOP. The current point $\boldsymbol{x}^{(t+1)}$ is optimal or near optimal.*
   - *Otherwise, reduce μ, increment $t \leftarrow t+1$ and go to step 1.*

The derivation of the move direction in step 1, and the value $1/\mu$ for the step size are conclusions of the second-order method utilized to solve this nonlinear problem. Then, since we wish to remain in the feasible region, we choose the smallest step size value between the obtained by the nonlinear method, and the smallest value that would take our solution outside the feasible region.

The rest of the algorithm makes use of what we know about μ. It is convenient to start with a large μ to make better progress in the interior of the feasible region. However, when we believe we are close to an optimal solution, we should decrease its value to get even closer.

7.5 Primal-dual interior-point search

In this section we use what we have learned from primal-dual simplex to obtain an interior-point method. Similarly to the derivation of primal-dual simplex, we work with a minimizing LP in standard form and its dual. This time, we also write the dual in standard form. Specifically, we work with the following problems:

$$(P)\quad \begin{aligned} \min\quad & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.}\quad & A\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned} \qquad\qquad (D)\quad \begin{aligned} \max\quad & \boldsymbol{b}^T\boldsymbol{\nu} \\ \text{s.t.}\quad & A^T\boldsymbol{\nu} + \boldsymbol{s} = \boldsymbol{c} \\ & \boldsymbol{s} \geq \boldsymbol{0} \end{aligned}$$

where $\boldsymbol{x}$ are the primal variables, $\boldsymbol{\nu}$ are the dual variables, and $\boldsymbol{s}$ are dual slack variables.

Motivated by the optimality conditions we used in dual-simplex algorithm, we obtain the following conditions in this case.

Principle 7.4 (Principle 7.22 from the textbook)**.** *The primal and dual solutions $\overline{\boldsymbol{x}}$ and $(\overline{\boldsymbol{\nu}}, \overline{\boldsymbol{s}})$ are optimal in their respective problems, if they satisfy the following conditions:*

(i) Primal feasibility: $A\overline{\boldsymbol{x}} = \boldsymbol{b}$ and $\overline{\boldsymbol{x}} \geq \boldsymbol{0}$

(ii) Dual feasibility: $A^T\boldsymbol{\nu} + \overline{\boldsymbol{s}} = \boldsymbol{c}$ and $\overline{\boldsymbol{s}} \geq \boldsymbol{0}$

(iii) Complementary slackness: $\overline{x}_j\overline{s}_j = 0$ for all j

Although the name of the algorithm is the same as in the variation of simplex, in this case the strategy is different. In primal-dual interior point search, we maintain primal and dual **strict** feasibility while seeking for complementary slackness.

Unless the primal and dual solutions are optimal, the complementary slackness condition will be violated. In other words, unless the primal and dual solutions $\overline{\boldsymbol{x}}$ and $(\overline{\boldsymbol{\nu}}, \overline{\boldsymbol{s}})$ are optimal, the **duality gap** $\boldsymbol{c}^T\overline{\boldsymbol{x}} - \boldsymbol{b}^T\overline{\boldsymbol{\nu}}$ is strictly positive. Further, we have the following principle.

Principle 7.5. *Suppose that $\overline{\boldsymbol{x}}$ is primal feasible, and $(\overline{\boldsymbol{\nu}}, \overline{\boldsymbol{s}})$ is dual feasible. Then, duality gap represents the total violation of the complementary slackness property.*

The proof is simple, and just needs rearrangement of terms, so we do it below:

Proof. By definition, the duality gap is $\boldsymbol{c}^T\overline{\boldsymbol{x}} - \boldsymbol{b}^T\overline{\boldsymbol{\nu}}$.

Now, from dual feasibility, we know $\boldsymbol{c} = A^T\overline{\boldsymbol{\nu}} + \overline{\boldsymbol{s}}$. Using this property in the duality gap, we obtain

$$\begin{aligned} \boldsymbol{c}^T\overline{\boldsymbol{x}} - \boldsymbol{b}^T\overline{\boldsymbol{\nu}} &= \left(A^T\overline{\boldsymbol{\nu}} + \overline{\boldsymbol{s}}\right)^T\overline{\boldsymbol{x}} - \boldsymbol{b}^T\overline{\boldsymbol{\nu}} \\ &= \overline{\boldsymbol{\nu}}^TA\overline{\boldsymbol{x}} + \overline{\boldsymbol{s}}^T\overline{\boldsymbol{x}} - \boldsymbol{b}^T\overline{\boldsymbol{\nu}} \qquad \text{(expanding the product of the first term)} \\ &= \overline{\boldsymbol{\nu}}^T\boldsymbol{b} + \overline{\boldsymbol{s}}^T\overline{\boldsymbol{x}} - \boldsymbol{b}^T\overline{\boldsymbol{\nu}} \qquad \text{(since } \overline{\boldsymbol{x}} \text{ is primal feasible)} \\ &= \overline{\boldsymbol{s}}^T\overline{\boldsymbol{x}} \qquad \text{(since } \overline{\boldsymbol{\nu}}^T\boldsymbol{b} = \boldsymbol{b}^T\overline{\boldsymbol{\nu}} \text{ because dot product is commutative)} \\ &= \sum_j \overline{x}_j\overline{s}_j \end{aligned}$$

that is, the duality gap equals the total violation of the complementary slackness property. □

In other words, this algorithm maintains strict primal and dual feasibility while decreasing the duality gap.

As usual, we need to determine improving directions and the step size. Since we are searching solutions in the interior of the feasible region, we seek directions $(\Delta\boldsymbol{x}, \Delta\boldsymbol{\nu}, \Delta\boldsymbol{s})$ that satisfy

$$A\Delta\boldsymbol{x} = \boldsymbol{0}, \quad A^T\Delta\boldsymbol{\nu} + \Delta\boldsymbol{s} = \boldsymbol{0}$$

A direction is improving if it reduces the duality gap. In this algorithm, we set a target $\mu > 0$ for the duality gap and we solve for $(\Delta\boldsymbol{x}, \Delta\boldsymbol{\nu}, \Delta\boldsymbol{s})$ so that the new point $(\boldsymbol{x} + \Delta\boldsymbol{x}, \boldsymbol{\nu} + \Delta\boldsymbol{\nu}, \boldsymbol{s} + \Delta\boldsymbol{s})$ has duality gap equal to μ. That is, a direction $(\Delta\overline{\boldsymbol{x}}, \Delta\overline{\boldsymbol{\nu}}, \Delta\overline{\boldsymbol{s}})$ at the current point $(\overline{\boldsymbol{x}}, \overline{\boldsymbol{\nu}}, \overline{\boldsymbol{s}})$ must satisfy:

$$\begin{aligned} & (\overline{x}_j + \Delta\overline{x}_j)(\overline{s}_j + \Delta\overline{s}_j) = \mu \qquad \forall j \\ \iff \quad & \overline{x}_j\overline{s}_j + \overline{x}_j\Delta\overline{s}_j + \Delta\overline{x}_j\overline{s}_j + \Delta\overline{x}_j\Delta\overline{s}_j = \mu \qquad \forall j \end{aligned}$$

The above equation is nonlinear because in the last term we have $\Delta\overline{x}_j\Delta\overline{s}_j$ and both are unknowns. This algorithm simply ignores this term to solve. Then, the direction $(\Delta\overline{\boldsymbol{x}}, \Delta\overline{\boldsymbol{\nu}}, \Delta\overline{\boldsymbol{s}})$ at the current point $(\overline{\boldsymbol{x}}, \overline{\boldsymbol{\nu}}, \overline{\boldsymbol{s}})$ satisfies:

$$\begin{aligned} &A\Delta\overline{\boldsymbol{x}} = \mathbf{0}, \\ &A^T\Delta\overline{\boldsymbol{\nu}} + \Delta\overline{\boldsymbol{s}} = \mathbf{0}, \\ &\overline{x}_j\overline{s}_j + \overline{x}_j\Delta\overline{s}_j + \Delta\overline{x}_j\overline{s}_j = \mu \quad \forall j \end{aligned}$$

The step size is chosen so that the next solution is both, primal and dual feasible. Hence, we obtain the following algorithm.

Algorithm 7.3 (Primal-dual interior point search)**.** *Starting from a minimizing primal and its dual, both in standard form as specified in problems (P) and (D) above,*

0. ***Initialization:***

0.1. Choose a strictly feasible primal solution $\boldsymbol{x}^{(0)}$ and a strictly feasible dual solution $(\boldsymbol{\nu}^{(0)}, \boldsymbol{s}^{(0)})$

0.2. Select target reduction factor $\rho \in (0, 1)$

0.3. Initialize duality gap: $g_0 \leftarrow \boldsymbol{c}^T\boldsymbol{x}^{(0)} - \boldsymbol{b}^T\boldsymbol{\nu}^{(0)}$

0.4. Compute the initial average complementarity target: $\mu_0 \leftarrow \dfrac{g_0}{n}$, where n is the number of primal variables

0.5. Initialize index $t \leftarrow 0$.

1. ***Optimality:***

- *If g_t is sufficiently close to 0, STOP. Current solutions $\boldsymbol{x}^{(t)}$, $\boldsymbol{\nu}^{(t)}$, $\boldsymbol{s}^{(t)}$ is optimal or nearly optimal.*
- *Otherwise, reduce $\mu_{t+1} \leftarrow \rho\mu_t$ and proceed to step 2.*

2. ***Move direction:***
Compute $\Delta\boldsymbol{x}^{(t+1)}, \Delta\boldsymbol{\nu}^{(t+1)}, \Delta\boldsymbol{s}^{(t+1)}$ by solving the system of equations:

$$\begin{aligned} &A\Delta\boldsymbol{x}^{(t+1)} = \mathbf{0} \\ &A^T\Delta\boldsymbol{\nu}^{(t+1)} + \Delta\boldsymbol{s}^{(t+1)} = \mathbf{0} \\ &x_j^{(t)}\Delta s_j^{(t+1)} + s_j^{(t)}\Delta x_j^{(t+1)} = \mu_{t+1} - x_j^{(t)}s_j^{(t)} \qquad \forall j \end{aligned}$$

3. ***Step size:***
Compute

$$\lambda \leftarrow \delta \min\{\lambda_P, \lambda_D\}$$

where

$$\lambda_P \leftarrow \min\left\{ \frac{x_j^{(t)}}{-\Delta x_j^{(t+1)}} : \Delta x_j^{(t+1)} < 0 \right\}$$

$$\lambda_D \leftarrow \min\left\{ \frac{s_j^{(t)}}{-\Delta s_j^{(t+1)}} : \Delta s_j^{(t+1)} < 0 \right\}$$

4. ***Move to next solution:***
Update solutions

$$\begin{aligned} \boldsymbol{x}^{(t+1)} &\leftarrow \boldsymbol{x}^{(t)} + \lambda\Delta\boldsymbol{x}^{(t+1)} \\ \boldsymbol{\nu}^{(t+1)} &\leftarrow \boldsymbol{\nu}^{(t)} + \lambda\Delta\boldsymbol{\nu}^{(t+1)} \\ \boldsymbol{s}^{(t+1)} &\leftarrow \boldsymbol{s}^{(t)} + \lambda\Delta\boldsymbol{s}^{(t+1)} \end{aligned}$$

advance $t \leftarrow t + 1$ and return to step 1.

Let's solve a short example.

Example 7.4 (based on Example 7.14 from the textbook)**.** *Consider the following LP*

$$\begin{aligned} \min \quad & 11x_1 + 5x_2 + 8x_3 + 16x_4 \\ s.t. \quad & -x_2 + 2x_3 + x_4 = 1 \\ & 2x_1 + x_2 + 3x_4 = 7 \\ & x_i \geq 0 \qquad \forall i \end{aligned}$$

(a) Compute the dual of the LP and place it in standard form.

(b) Show that the primal-dual interior point algorithm could start with $\boldsymbol{x} = (1, 2, 1, 1)$ and $\boldsymbol{\nu} = (3, 4)$

(c) Compute the duality gap and show that it is equivalent to the total violation of complementary slackness.

(d) Write the system of equations that will need to be solved in step 2 using a shringking factor of $\rho = 0.6$.

Solution.

(a) We first write the dual, and then we place it in standard form. Assigning the dual variables ν_1 and ν_2 to the constraints, we obtain

$$\begin{aligned} \max \quad & \nu_1 + 7\nu_2 \\ \text{s.t.} \quad & 2\nu_2 \leq 11 \\ & -\nu_1 + \nu_2 \leq 5 \\ & 2\nu_1 \leq 8 \\ & \nu_1 + 3\nu_2 \leq 16 \\ & \nu_i \text{ unrestricted} \quad \forall i \end{aligned}$$

Adding slack variables to place it in standard form we obtain

$$\begin{aligned} \max \quad & \nu_1 + 7\nu_2 \\ \text{s.t.} \quad & 2\nu_2 + s_1 = 11 \\ & -\nu_1 + \nu_2 + s_2 = 5 \\ & 2\nu_1 + s_3 = 8 \\ & \nu_1 + 3\nu_2 + s_4 = 16 \\ & s_i \geq 0 \qquad \forall i \end{aligned}$$

(b) Observe that $\boldsymbol{x} > \mathbf{0}$, that is, all its elements are strictly positive. Then, we only need to check if it satisfies the constraint to conclude that it is an interior point. We plug the numbers on the left-hand side of each of the constraints and obtain:

$$\begin{aligned} -2 + 2 \cdot 1 + 1 &= 1 \\ 2 \cdot 1 + 2 \cdot 1 + 3 \cdot 1 &= 7 \end{aligned}$$

Hence, $\boldsymbol{x}$ is an interior point of the primal feasible region.

To check if $\boldsymbol{\nu}$ is an interior point, we compute the slack variables $\boldsymbol{s}$ and we check whether $\boldsymbol{s} > \mathbf{0}$. We obtain

$$\begin{aligned} 2\nu_2 = 8 \quad &\Longrightarrow \quad s_1 = 3 \\ -\nu_1 + \nu_2 = 1 \quad &\Longrightarrow \quad s_2 = 4 \\ 2\nu_1 = 6 \quad &\Longrightarrow \quad s_3 = 2 \\ \nu_1 + 3\nu_2 = 15 \quad &\Longrightarrow \quad s_4 = 1 \end{aligned}$$

Since $s_i > 0$ for all i, we conclude that $(\boldsymbol{\nu}, \boldsymbol{s})$ is an interior point of the dual feasible region.

(c) We first compute the duality gap. We obtain:

$$\boldsymbol{c}^T\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{\nu} = 11 \cdot 1 + 5 \cdot 2 + 8 \cdot 1 + 16 \cdot 1 - (3 + 7 \cdot 4) = 45 - 31 = 14$$

The total violation of complementary slackness is:

$$\boldsymbol{x}^T\boldsymbol{s} = \sum_{j=1}^{4} x_j s_j = 1 \cdot 3 + 2 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 = 14$$

Then, we have $\boldsymbol{c}^T\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{\nu} = \boldsymbol{x}^T\boldsymbol{s}$.

(d) We write each of the three sets of equations:

- $A\Delta\boldsymbol{x} = \mathbf{0}$:

$$\begin{aligned} -\Delta x_2 + 2\Delta x_3 + \Delta x_4 &= 0 \\ 2\Delta x_1 + \Delta x_2 + 3\Delta x_4 &= 0 \end{aligned}$$

- $A^T\Delta\boldsymbol{\nu} + \Delta\boldsymbol{s} = \mathbf{0}$:

$$\begin{aligned} 2\Delta\nu_2 + \Delta s_1 &= 0 \\ -\Delta\nu_1 + \Delta\nu_2 + \Delta s_2 &= 0 \\ 2\Delta\nu_1 + \Delta s_3 &= 0 \\ \Delta\nu_1 + 3\Delta\nu_2 + \Delta s_4 &= 0 \end{aligned}$$

- $x_j\Delta s_j + \Delta x_j s_j = \rho\mu - x_j s_j \qquad \forall j$

$$\begin{aligned} \Delta s_1 + 3\Delta x_1 &= 0.6 \cdot \frac{14}{4} - 1 \cdot 3 \\ 2\Delta s_2 + 4\Delta x_2 &= 0.6 \cdot \frac{14}{4} - 2 \cdot 4 \\ \Delta s_3 + 2\Delta x_3 &= 0.6 \cdot \frac{14}{4} - 1 \cdot 2 \end{aligned}$$

$$\Delta s_4 + \Delta x_4 = 0.6 \cdot \frac{14}{4} - 1 \cdot 1$$

Computing the numbers on the right-hand side we obtain

$$\begin{aligned}
\Delta s_1 + 3\Delta x_1 &= -\frac{9}{10} \\
2\Delta s_2 + 4\Delta x_2 &= -\frac{59}{10} \\
\Delta s_3 + 2\Delta x_3 &= \frac{1}{10} \\
\Delta s_4 + \Delta x_4 &= \frac{11}{10}
\end{aligned}$$

□

8 Discrete Optimization Models

[This section is based on Chapter 11 of the textbook]

The entire semester, we have been discussing the importance of linear problems and the wide variety of situations we can model. In this chapter we will expand the set of problems we can model by introducing integer variables. We are particularly interested in binary variables (that is, variables that take the value 0 or 1) because they open a whole new world of objective functions and constraints to model. In this chapter we will go through some of the most well-used examples.

In this chapter we will construct problems that have both, continuous and integer variables. However, besides the integer variable-type constraint, they are linear. Hence, they are called mixed-integer programming (MIP) models.

8.1 Either/or constraints

The first type of models we will learn are models that are completely linear, except by a constraint or the objective function, which can be of two or more types. That is, we model either/or statements. Whithin this cathegory, we will see two types of problems to model.

All or nothing constraints

We are interested in modeling situations where we have an indivisible resource. That is, if we decide to use it, we must use it all.

Principle 8.1 (based on Principle 11.1 from the textbook)**.** *All-or-nothing variable requirements of the form $x_j = 0$ or $x_j = u_j$ can be modeled with an additional binary variable y_j defined as*

$$y_j = \begin{cases} 1 & \text{, if we decide } x_j = u_j \\ 0 & \text{, otherwise} \end{cases}$$

and adding the MIP constraint

$$x_j = y_j u_j$$

The variable y_j can be interpreted as the fraction of u_j that we decide to take: all or nothing.

Let's see an example.

Example 8.1 (Example 11.1 from the textbook)**.** *Consider the following linear program*

$$\begin{aligned} \max \quad & 18x_1 + 3x_2 + 9x_3 \\ s.t. \quad & 2x_1 + x_2 + 7x_3 \leq 150 \\ & 0 \leq x_1 \leq 60 \\ & 0 \leq x_2 \leq 30 \\ & 0 \leq x_3 \leq 20 \end{aligned}$$

Revise the model so that each variable can be used only at zero or its upper bound.

Solution. We introduce the following variables:

$$y_j = \begin{cases} 1 & \text{if we decide to use the whole resource availability of } x_j \\ 0 & \text{otherwise} \end{cases} \qquad \forall j$$

Then, we may modify the problem in the two following ways:

- **Method 1**:

$$\begin{aligned} \max \quad & 18x_1 + 3x_2 + 9x_3 \\ \text{s.t.} \quad & 2x_1 + x_2 + 7x_3 \leq 150 \\ & x_1 = 60y_1 \\ & x_2 = 30y_2 \\ & x_3 = 20y_3 \\ & y_j \in \{0,1\} \qquad \forall j \end{aligned}$$

  Here we kept the model as is, and we added constraints that establish that x_j can take the value of the upper bound or zero.

- **Method 2:**

$$\begin{aligned} \max \quad & 18(60y_1) + 3(30y_2) + 9(20y_3) \\ \text{s.t.} \quad & 2(60y_1) + 30y_2 + 7(20y_3) \leq 150 \\ & y_j \in \{0,1\} \qquad \forall j \end{aligned}$$

  Here we rewrite the problem in terms of y_j, using the definition of y_j in terms of x_j and the upper bounds. Multiplying the numbers, we obtain

$$\begin{aligned} \max \quad & 1080y_1 + 90y_2 + 180y_3 \\ \text{s.t.} \quad & 120y_1 + 30y_2 + 140y_3 \leq 150 \\ & y_j \in \{0,1\} \qquad \forall j \end{aligned}$$

□

Fixed costs

In many real-life situations, the cost of using a resource has a fixed and a variable component. For example, there might be a setup cost for using a machine, or for renting a storage room. Additionally, we must pay for each unit processed by the machine, or each unit of inventory stored in the storage room.

In general, we may model the cost of a nonnegative variable x as

$$\theta(x) = \begin{cases} f + cx & \text{if } x > 0 \\ 0 & \text{if } x = 0 \end{cases}$$

Principle 8.2 (based on Principle 11.2)**.** *Minimize objective functions with a cost structure as $\theta(x)$ above can me modeled by introducing new fixed-cost variables*

$$y_j = \begin{cases} 1 & \text{if } x_j > 0 \\ 0 & \text{if } x_j = 0 \end{cases}$$

Then, the objective function becomes

$$\min \quad fy_j + cx_j$$

The principle above tells us how to include a mixed cost structure in the objective function. However, the variable y_j was only defined in English and we need to explicitly add it to the model. In other words, if we solve the model above with some software, the software will not understand our definition. We need to add constraints that will imply the definition that we gave. We do that by adding switching constraints.

Definition 8.1 (Definition 11.3 from the textbook)**.** *Switching constraints model the requirement that a continuous variable $x_j > 0$ can be used only if a corresponding binary variable $y_j = 1$ by*

$$x_j \leq u_j y_j$$

where u_j is an upper bound to the value of x_j. Such an upper bound can be given explicitly, or it can be implied by the other constraints of the problem.

A switching constraint models the following two statements at the same time:

- If $x_j > 0$, then $y_j = 1$
- If $y_j = 0$, then $x_j = 0$ as well.

Hence, it models the relationship we want between the fixed-cost variable y_j and the variable-cost variable x_j. Let's see some examples.

Example 8.2 (Example 11.2 from the textbook)**.** *Consider the following objective function*

$$\min \quad \theta_1(x_1) + \theta_2(x_2)$$

where

$$\theta_1(x_1) = \begin{cases} 150 + 7x_1 & \text{, if } x_1 > 0 \\ 0 & \text{, if } x_1 = 0 \end{cases} \quad \text{and} \quad \theta_2(x_2) = \begin{cases} 110 + 9x_2 & \text{, if } x_2 > 0 \\ 0 & \text{, if } x_2 = 0 \end{cases}$$

For each of the following sets of constraints on x_1 and x_2, provide a MIP.

(a) $x_1 + x_2 \geq 8$
$0 \leq x_1 \leq 3$
$0 \leq x_2 \leq 8$

(b) $x_1 + x_2 \geq 8$
$2x_1 + x_2 \leq 10$
$x_1, x_2 \geq 0$

Solution. In both cases, we introduce the following binary decision variables:

$$y_j = \begin{cases} 1 & \text{if } x_j > 0 \\ 0 & \text{if } x_j = 0 \end{cases} \qquad \forall j \in \{1, 2\}$$

(a) Observe that we have explicit bounds for x_1 and x_2. Hence, we use them for the switching constraints and we obtain the following MIP:

$$\begin{aligned} \min \quad & 7x_1 + 9x_2 + 150y_1 + 110y_2 \\ \text{s.t.} \quad & x_1 + x_2 \geq 8 \\ & x_1 \leq 3y_1 \\ & x_2 \leq 8y_2 \\ & x_1, x_2 \geq 0 \\ & y_1, y_2 \in \{0, 1\} \end{aligned}$$

(b) Here we do not have explicit bounds, but observe that the second constraint implies that $x_1 \leq 5$ and $x_2 \leq 10$. Hence, we use these numbers as upper bounds. We obtain

$$\begin{aligned} \min \quad & 7x_1 + 9x_2 + 150y_1 + 110y_2 \\ \text{s.t.} \quad & x_1 + x_2 \geq 8 \\ & 2x_1 + x_2 \leq 10 \\ & x_1 \leq 5y_1 \\ & x_2 \leq 10y_2 \\ & x_1, x_2 \geq 0 \\ & y_1, y_2 \in \{0, 1\} \end{aligned}$$

□

8.2 Knapsack and budgeting constraints

In this type of problems, we have a long list of items that we can choose and a budget that prevents us from choosing them all. The goal is to decide which items to choose in order to maximize our happiness.

Principle 8.3. *In knapsack and budgeting problems, we use integer variables defined as*

$$y_j = \begin{cases} 1 & \text{if we choose item } j \\ 0 & \text{if not} \end{cases}$$

Observe that in this type of problems, all the variables are integer. Let's see an example.

Example 8.3 (Application 11.1 from the textbook)**.** *Indy Car wants to improve their cars' features for this year's race. Six different features may be added to each car. An estimation of the cost and the speed enhancement is presented in the following table*

	Feature j					
	1	*2*	*3*	*4*	*5*	*6*
Cost (in thousand USD)	*10.2*	*6*	*23*	*11*	*9.8*	*31.6*
Speed increase (mph)	*8*	*3*	*15*	*7*	*10*	*12*

Suppose that Indy Car wants to maximize the performance gain without exceeding a budget of \$35K. Formulate a linear integer program to solve Indy Car's problem.

Solution. We use the following variables:

$$y_j = \begin{cases} 1 & \text{if feature } j \text{ is added to the racing cars} \\ 0 & \text{otherwise} \end{cases} \qquad \forall j \in \{1, 2, 3, 4, 5, 6\}$$

Then, we obtain the following integer programming model:

$$\begin{aligned} \max \quad & 8y_1 + 3y_2 + 15y_3 + 7y_4 + 10y_5 + 12y_6 \\ \text{s.t.} \quad & 10.2y_1 + 6y_2 + 23y_3 + 11y_4 + 9.8y_5 + 31.6y_6 \leq 35 \\ & y_j \in \{0, 1\} \qquad \forall j \in \{1, 2, 3, 4, 5, 6\} \end{aligned}$$

□

In the problem above, our decision variables were whether we choose feature j or not. However, in practice we may also face problems where we need to decide the number of features/items to include in our knapsack. Let's see an example.

Example 8.4 (Example 11.3 from the textbook)**.** *Readily available US coins are denominated 1, 5, 10, and 25 cents. Formulate an integer programming model to minimize the number of coins needed to provide change amount of exactly q cents.*

Solution. We use the decision variables:

$$y_j = \text{ Number of coins type } j \text{ to use} \qquad \forall j \in \{1, 5, 10, 25\}$$

Then, we obtain the following model

$$\begin{aligned} \min \quad & y_1 + y_5 + y_{10} + y_{25} \\ \text{s.t.} \quad & y_1 + 5y_5 + 10y_{10} + 25y_{25} = q \\ & y_j \geq 0 \text{ and integer} \qquad \forall j \end{aligned}$$

□

Satisfying multiple budget constraints

In the examples above, there is just one budget constraint. However, sometimes we might have multiple periods and they may have different budget requirements, as shown in the following example.

Example 8.5 (Example 11.4 from the textbook)**.** *A department store is considering 4 possible expansions into presently unoccupied space in a shopping mall. The following table shows how much (in millions of dollars) each expansion would cost in the next two fiscal years, and the required floor space (in thousands of square feet). Additionally, it shows the budget of each year (in millions of dollars) and the available space for expansions (in thousands of square feet).*

	Expansion option j				*Budget*
	1	*2*	*3*	*4*	
Year 1	*1.5*	*5.0*	*7.3*	*1.9*	*10*
Year 2	*3.5*	*1.8*	*6.0*	*4.2*	*10*
Space	*2.2*	*9.1*	*5.3*	*8.6*	*17*

Formulate constraints on investment funds and floor space, assuming that the objective function is well-known (and, hence, irrelevant to us).

Solution. Observe that in this problem we must decide which expansion option to use once, and this decision will have consequences for 2 years. Then, we use the decision variables:

$$y_j = \begin{cases} 1 & \text{if expansion } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

and we write constraints for each year's budget and for the available space. We obtain

$$\begin{aligned} 1.5y_1 + 5y_2 + 7.3y_3 + 1.9y_4 &\leq 10 \quad \text{(budget of year 1)} \\ 3.5y_1 + 1.8y_2 + 6.0y_3 + 4.2y_4 &\leq 10 \quad \text{(budget of year 2)} \\ 2.2y_1 + 9.1y_2 + 5.3y_3 + 8.6y_4 &\leq 17 \quad \text{(available space)} \end{aligned}$$

□

Modeling mutually exclusive decisions

When we started this section, we talked about the number of items we can fit to satisfy a budget constraint. If the choice of items is mutually exclusive, it means that we can choose at most one item from the list and, hence, the budget is 1. Let's see an example.

Example 8.6 (Example 11.5 from the textbook)**.** *Suppose that a real estate development company is considering 5 investment decisions. Only one of the first 3 can be chosen because all require the same piece of land. Investment 4 is a new office building, and investment 5 is the same project delayed a year, so only one of them can be chosen too.*

Formulate constraints to model these decisions.

Solution. In this case we are only asked to formulate the constraints, not the objective function. We use the variables

$$y_j = \begin{cases} 1 & \text{if investment } j \text{ is chosen} \\ 0 & \text{otherwise} \end{cases} \qquad \forall j \in \{1, 2, 3, 4, 5\}$$

Then, the constraint that only one of the first three investments can be chosen is

$$y_1 + y_2 + y_3 \leq 1$$

and the constraint that only one between 4 and 5, is

$$y_4 + y_5 \leq 1$$

□

Modeling dependency between choices

Another typical constraint in budgeting is that two or more projects are linked together. That is, choosing a certain project implies that we must also choose another one. Let's see an example.

Example 8.7 (Example 11.6 from the textbook)**.** *Phase 1 of a new city hall project will construct the first story of a building that could grow to two stories in phase 2, and to three in phase 3. Model the dependency among phases with binary variables and linear constraints.*

Solution. The dependency is as follows:

- If we build phase j, we may or may not build phase $j + 1$ (for $j = 1, 2$)
- If we decide to build phase $j + 1$, then we must have built phase j (for $j = 1, 2$)

Then, we use the decision variables

$$y_j = \begin{cases} 1 & \text{if phase } j \text{ is constructed} \\ 0 & \text{otherwise} \end{cases} \qquad \forall j \in \{1,2,3\}$$

and we obtain the following constraints

$$y_2 \leq y_1$$
$$y_3 \leq y_2$$

□

The example above shows the most basic implication: If I choose a certain item, I must choose another one. However, we can also model more complicated implications, as in the example below:

Example 8.8. *Use binary variables to model the following if/else statements:*

(a) If $x_1 + x_2 > 3$, then $3x_1 - 2x_2 \leq 5$

(b) If $x_1 + 2x_2 < 5$, then $x_1 + 3x_2 \geq 3$

Solution. In both cases, we add an auxiliary binary variable and we use switching constraints. In the switching constraints we use a parameter M to make sure that we do not impose additional constraints while modeling the if/else statement.

(a) We add the variable

$$y = \begin{cases} 1 & \text{if } x_1 + x_2 > 3 \\ 0 & \text{otherwise} \end{cases}$$

and we model the if/else statement as follows:

$$\begin{aligned} x_1 + x_2 &\leq 3 + My \qquad (\text{if } x_1 + x_2 > 3, \text{ then } y = 1) \\ 3x_1 - 2x_2 &\leq 5 + M(1-y) \qquad (\text{if } y = 1, \text{ then } 3x_1 - 2x_2 \leq 5) \\ y &\in \{0,1\} \end{aligned}$$

(b) Similarly to the previous part, we define a binary variable that equals 1 if the "if" statement is true, as follows:

$$y = \begin{cases} 1 & \text{if } x_1 + 2x_2 < 5 \\ 0 & \text{otherwise} \end{cases}$$

Then,

$$\begin{aligned} x_1 + 2x_2 &\geq 5 - My \qquad (\text{if } x_1 + 2x_2 < 5, \text{ then } y = 1) \\ x_1 + 3x_2 &\geq 3 - M(1-y) \qquad (\text{if } y = 1, \text{ then } x_1 + 3x_2 \geq 3) \\ y &\in \{0,1\} \end{aligned}$$

□

8.3 Set packing, Covering and Partitioning Models

Set packing, covering and partitioning models can be used when we need to establish how many of the available resources must be chosen from a given subset. Specifically, we have the following definitions:

Definition 8.2 (Definitions 11.9-11.11 from the textbook)**.** *Suppose that we are interested in writing set packing, covering and partitioning constraints for a subset of variables. This subset is characterized by the set of indices J. Define the decision variables:*

$$y_j = \begin{cases} 1 & \textit{if resource } j \textit{ is selected} \\ 0 & \textit{if not} \end{cases} \qquad \forall j \in J$$

Then,

- ***Set covering** constraints model when we need to choose **at least** 1 resource from the set J, and are expressed as*

$$\sum_{j \in J} y_j \geq 1$$

- ***Set packing** constraints model when we need to choose **at most** 1 resource from the set J, and are expressed as*

$$\sum_{j \in J} y_j \leq 1$$

- ***Set partitioning** constraints model when we need to choose **exactly** 1 resource from the set J, and are expressed as*

$$\sum_{j \in J} y_j = 1$$

Let's see an example.

Example 8.9 (Example 11.7 from the textbook)**.** *A university is acquiring mathematical programming software for use in operations research classes. The four codes available, and the types of optimization algorithms they provide are indicated by ✓ in the following table. Additionally, objective function coefficients are provided and in each of the following tasks we specify what they mean.*

Algorithm type	*Code j*			
	1	*2*	*3*	*4*
Linear Programming (LP)	✓	✓	✓	✓
Integer Programming (IP)		✓		✓
Nonlinear Programming (NLP)			✓	✓
Objective parameters	*3*	*4*	*6*	*14*

Provide a MIP for the following goals.

(a) If the objective coefficients represent cost, formulate a MIP to acquire a minimum cost collection of codes providing LP, IP and NLP capability

(b) If the objective coefficients represent cost, formulate a MIP to acquire a minimum cost collection of codes with exactly one providing LP, one IP and one NLP

(c) If the objective coefficients represent quality of the code, formulate a MIP to acquire a maximum quality collection of codes with at most one providing each of the three algorithm types.

Solution. In all cases, we use the decision variables

$$y_j = \text{ if code } j \text{ is chosen} \qquad \forall j \in \{1,2,3,4\}$$

(a) We need to make sure that each of the three algorithm types is covered by at least one of the acquired softwares. Then, we obtain the following model

$$\begin{aligned} \min \quad & 3y_1 + 4y_2 + 6y_3 + 14y_4 \\ \text{s.t.} \quad & y_1 + y_2 + y_3 + y_4 \geq 1 \qquad \text{(LP)} \\ & y_2 + y_4 \geq 1 \qquad \text{(IP)} \\ & y_3 + y_4 \geq 1 \qquad \text{(NLP)} \\ & y_j \in \{0,1\} \qquad \forall j \end{aligned}$$

(b) In this case we need only one software for each algorithm type, so we obtain

$$\begin{aligned} \min \quad & 3y_1 + 4y_2 + 6y_3 + 14y_4 \\ \text{s.t.} \quad & y_1 + y_2 + y_3 + y_4 = 1 \qquad \text{(LP)} \\ & y_2 + y_4 = 1 \qquad \text{(IP)} \\ & y_3 + y_4 = 1 \qquad \text{(NLP)} \\ & y_j \in \{0,1\} \qquad \forall j \end{aligned}$$

(c) In this case, we maximize the objective function (because the coefficients mean quality) and we make sure that at most one code provides each algorithm type. We obtain

$$\begin{aligned} \max \quad & 3y_1 + 4y_2 + 6y_3 + 14y_4 \\ \text{s.t.} \quad & y_1 + y_2 + y_3 + y_4 \leq 1 \qquad \text{(LP)} \\ & y_2 + y_4 \leq 1 \qquad \text{(IP)} \\ & y_3 + y_4 \leq 1 \qquad \text{(NLP)} \\ & y_j \in \{0,1\} \qquad \forall j \end{aligned}$$

□

Theoretically, we can ask for anything with these set constraints. However, we may easily obtain infeasible problems. A possible solution to this (very common) issue is to relax the problem by allowing to break some of the set constraints and assigning a cost for each one we break. Let's see this modification in action with our softwares example.

Example 8.10 (Example 11.8 from the textbook)**.** *In the software example, suppose that we are interested maximizing the number of algorithms available under a budget of 12.*

Solution. We can model this problem as a set covering model with the variables y_j defined before, and

$$z_i = \begin{cases} 1 & \text{if algorithm type } i \text{ is not covered} \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in \{LP, IP, NLP\}$$

Observe that maximizing the number of covered algorithms is equivalent to minimizing the number of uncovered algorithms. Then, we obtain the following model

$$\begin{aligned} \min \quad & z_{LP} + z_{IP} + z_{NLP} \\ \text{s.t.} \quad & 3y_1 + 4y_2 + 6y_3 + 14y_4 \leq 12 \qquad \text{(budget)} \\ & y_1 + y_2 + y_3 + y_4 + z_{LP} \geq 1 \qquad \text{(LP)} \\ & y_2 + y_4 + z_{IP} \geq 1 \qquad \text{(IP)} \\ & y_3 + y_4 + z_{NLP} \geq 1 \qquad \text{(NLP)} \\ & y_j \in \{0,1\} \qquad \forall j \\ & z_i \in \{0,1\} \qquad \forall i \end{aligned}$$

□

8.4 Assignment and Matching Models

In assignment models, we usually have a list of tasks that need to be performed, and a list of people that can perform the task. Then, the goal is to match each person to one task and to make sure that every task is performed. Formally, we have the following definition.

Definition 8.3 (Definition 11.14 from the textbook)**.** *In an assignment problem, we use variables*

$$y_{i,j} = \begin{cases} 1 & \text{if person } i \text{ performs task } j \\ 0 & \text{otherwise} \end{cases} \qquad \forall i, j$$

and the assignment constraints are

$$\begin{aligned} \sum_i y_{i,j} &= 1 \qquad \forall j \\ \sum_j y_{i,j} &= 1 \qquad \forall i \\ y_{i,j} &\in \{0,1\} \qquad \forall i, j \end{aligned}$$

Let's see an example.

Example 8.11 (Example 11.10 from the textbook)**.** *A swimming coach is choosing his team for a medley relay. One swimmer will swim each leg $j \in \{1,2,3,4\}$ where:*

$$\begin{aligned} j = 1&: \quad \textit{Back stroke} \\ j = 2&: \quad \textit{Breast stroke} \\ j = 3&: \quad \textit{Butterfly stroke} \\ j = 4&: \quad \textit{Free style} \end{aligned}$$

From previous experience, the coach can estimate the time t_{ij} that swimmer i could achieve on leg j. Assuming that there are 4 available swimmers, formulate a MIP to choose the fastest medley team.

Solution. We use the decision variables:

$$y_{ij} = \begin{cases} 1 & \text{if swimmer } i \text{ swims leg } j \\ 0 & \text{otherwise} \end{cases}$$

Then, we obtain the following model:

$$\begin{aligned} \min \quad & \sum_{i=1}^{4}\sum_{j=1}^{4} t_{ij} y_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^{4} y_{i,j} = 1 \qquad \forall i \qquad \text{(each swimmer swims one leg)} \\ & \sum_{i=1}^{4} y_{i,j} = 1 \qquad \forall j \qquad \text{(each leg is swum by one swimmer)} \\ & y_{i,j} \in \{0,1\} \qquad \forall i, j \end{aligned}$$

□

A special case of assignment problems is when both sets of indices are the same, that is, when we are trying to pair up items of a set. Let's see an example.

Example 8.12 (Example 11.13 from the textbook)**.** *The instructor of a class is trying to create teams of two people for a final project. Each student s has scored their preference $p_{s,s'}$ to work with each other student $s' \neq s$.*

Formulate an IP model to maximize the total preference assuming that there are 20 students.

Solution. For $s, s' \in \{1, \ldots, 20\}$ with $s \neq s'$, we use the decision variables

$$y_{s,s'} = \begin{cases} 1 & \text{if student } s \text{ is teamed with student } s' \\ 0 & \text{otherwise} \end{cases}$$

Then, we obtain the following model:

$$\begin{aligned} \max \quad & \sum_{s=1}^{19}\sum_{s'=s+1}^{20} (p_{s,s'} + p_{s',s}) y_{s,s'} \\ \text{s.t} \quad & \sum_{s'=1}^{s-1} y_{s,s'} + \sum_{s'=s+1}^{20} y_{s,s'} = 1 \qquad \forall s \in \{2, \ldots, 19\} \\ & \sum_{s'=2}^{20} y_{1,s'} = 1 \end{aligned}$$

$$\sum_{s'=1}^{19} y_{20,s'} = 1$$
$$y_{s,s'} \in \{0,1\} \qquad \forall s, s' \in \{1,\dots,20\},\ s' > s$$

In the objective function, we are considering the "total happiness" of teaming up student s with s' by adding up the preference of both of them. Then, we add up the happiness of each team accross all possible teams. To avoid double counting, we write the sum so that $s' > s$.

In the constraint, each student must be paired with a different student, so we split the sum into $s' < s$ and $s' > s$. We write the extreme cases ($s = 1$ and $s = 20$) separately because they only need one of the summations. □

8.5 Traveling Salesperson and Routing Models

A special and very useful type of integer-programming models, are routing models. In these problems, we usually have a set of locations and we need to create the best possible route according to constraints that are specific to each case. The Traveling Salesperson problem is a specific routing problem, that we define below.

Definition 8.4 (Definition 11.23 from the textbook)**.** *The Traveling Salesperson Problem (TSP) seeks a minimum-length route visiting every location exactly once.*

For example, we can think of a person that needs to deliver a list of items. Each item must be delivered to a different address, so the person needs to visit each of these addresses once.

We will only cover the symmetric TSP, that is, where the cost of going from location i to j is equal to the cost of going from j to i. To provide a model, suppose that we have a set of nodes N that must be visited by the traveling salesperson, and for each pair $i, j \in N$, the cost of traveling between nodes i and j is $d_{i,j}$.

We use the following variables. For $i, j \in N$ with $i < j$, define

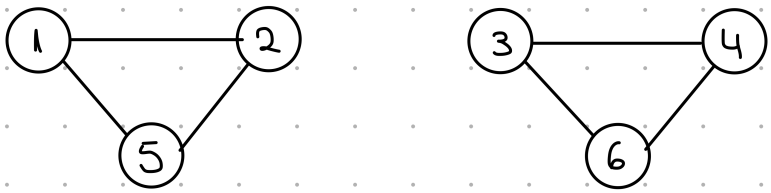
$$y_{ij} = \begin{cases} 1 & \text{if the route includes a leg between nodes } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$

Then, the following model visits every node once at minimum cost:

$$\begin{aligned} \min \quad & \sum_{i \in N} \sum_{j \in N:\, j>i} d_{ij} y_{ij} \\ \text{s.t.} \quad & \sum_{j<i} y_{j,i} + \sum_{j>i} y_{ij} = 2 \qquad \forall i \in N \\ & y_{ij} \in \{0,1\} \qquad \forall i, j \in N,\ j > i \end{aligned}$$

The objective function is the minimization of the route's cost, and the constraint establishes that we must enter and leave every node i. That way, we make sure we visit all the nodes and we do not get stuck in any of them.

However, this model is not complete. For example, if $N = \{1,\dots,6\}$ consider the following solution



where

$$y_{12} = y_{15} = y_{25} = y_{34} = y_{36} = y_{46} = 1$$
$$y_{ij} = 0 \qquad \text{for all other } i, j$$

satisfies all the constraints that we established above. However, the solution is not a route because it has two subtours.

Then, we need to add constraints that will prevent subtours. There are several ways to do it, but here we present one. The following formulation is complete for the TSP.

$$\begin{aligned} \min \quad & \sum_{i \in N} \sum_{j \in N:\, j>i} d_{ij} y_{ij} \\ \text{s.t.} \quad & \sum_{j<i} y_{j,i} + \sum_{j>i} y_{ij} = 2 \qquad \forall i \in N \\ & \sum_{i \in \mathcal{S}} \sum_{j \notin \mathcal{S},\, j>i} y_{i,j} + \sum_{i \notin \mathcal{S}} \sum_{j \in \mathcal{S},\, j>i} y_{i,j} \geq 2 \qquad \forall \mathcal{S} \subsetneq N \\ & y_{ij} \in \{0,1\} \qquad \forall i, j \in N,\ j > i \end{aligned}$$

The constraint we added says that for every proper subset of the nodes $\mathcal{S}$, the number of edges that connect nodes inside $\mathcal{S}$ with nodes outside $\mathcal{S}$ has to be at least 2. That is, the proposed routes must enter and leave every subset of nodes.

There is one big struggle about the subtour elimination constraints: the number of constraints increases exponentially with the number of nodes (delivery locations). Then, solving the TSP to optimality can take considerable amount of time. In practice, some people add the subtour elimination constraints as needed. That is, they start with the first model we provided and solve it. If the optimal solution has subtours, they only add the subtour-elimination constraints corresponding to those specific subtours. Then, they solve again and repeat.

8.6 Single Processor Scheduling

In this type of problems, we have a single machine that can process jobs that arrive at different moments in time and have a due date. The task is to find the best sequence to start jobs and meet the deadlines, assuming that the processor can work on one task at a time.

Let's use the following example to learn these models.

Example 8.13 (based on Application 11.12 from the textbook)**.** *Let's consider the binder scheduling problem confronting a copy shop, named Nifty Notes. Just before the start of the semester, professors at the nearby university supply Nifty Notes with a single original of their class handouts, a projection of the class enrollment and a due date by which copies should be available. Then, the Nifty Notes staff must rush to print and bind the required number of copies before each class begins.*

During the busy period each semester, Nifty Notes operates its single binding station 24 hours per day. The table below shows process times, release times, and due dates for the jobs $j = 1, 2, 3, 4$ now pending at the binder. Additionally, we present the cost of each 'late day' for each job, that is, the cost we must pay per day beyond the due date if the job is not ready.

Parameter	*Binder job j*					
	1	*2*	*3*	*4*	*5*	*6*
Process time, p_j	*12*	*8*	*3*	*10*	*4*	*18*
Release time, r_j	*0*	*5*	*8*	*10*	*17*	*22*
Due date, d_j	*30*	*22*	*92*	*12*	*21*	*80*
Cost per late day, c_j	*18*	*9*	*81*	*1*	*1*	*40*

Nifty notes wishes to minimize the total late cost. Formulate a linear integer program to model their scheduling problem, assuming that only one job can be processed at a time.

Solution. We first list our decision variables. For $j \in \{1, \ldots, 6\}$ define:

$$\begin{array}{lcl} x_j & = & \text{Time at which binding starts job } j \\ z_j & = & \text{Number of extra days used to finish job } j \\ y_{j,j'} & = & \text{Binary variable that equals 1 if job } j \text{ starts before job } j', \text{ for all } j' \neq j. \end{array}$$

Then, the objective function is

$$\min \quad \sum_{j=1}^{6} c_j z_j$$

We describe the constraints below:

- We cannot start processing a job before the release date:

$$x_j \geq r_j \qquad \forall j$$

- We can only process one job at a time. Then, the time interval when I am processing job j cannot be used by any other job j'. In the constraints below we model that either job j must be finished before j' or viceversa:

$$x_j + p_j \leq x_{j'} + M(1 - y_{j,j'}) \qquad \forall j, \forall j' \neq j$$
$$x_{j'} + p_{j'} \leq x_j + M y_{j,j'} \qquad \forall j, \forall j' \neq j$$

  The left-hand side represents when job j and j' (respectively) will be finished, and the right-hand side depends on whether job j goes before job j' or not. If job j goes before job j', then job j must be finished before we can start job j'. In such case, $y_{j,j'} = 1$ and the first constraint turns into

$$x_j + p_j \leq x_{j'}$$

  which explicitly says that job j must be finished before we can start job j'. When $y_{j,j'} = 1$, the second constraint becomes:

$$x_{j'} + p_{j'} \leq x_j + M$$

  and since M is a large constant, it is saying something like $x_{j'} + p_{j'} \leq \infty$, which is trivial.

  Now, if job j' goes before j, then $y_{j,j'} = 0$ and the first constraint becomes trivial. It is not the second constraint that explicitly models the order in which we process the jobs.

- Nifty notes needs to pay for each extra day they take beyond the due date. Then, we need a constraint that explicitly computes the value of z_j. We have:

$$x_j + p_j \leq d_j + z_j \qquad \forall j$$

  where the left-hand side represents the time at which job j can be finished, and the right-hand side represents the due date plus the extra days.

- The remaining constraints are the variable-type constraints:

$$\begin{aligned} & x_j \geq 0 \qquad \forall j \\ & z_j \geq 0 \qquad \forall j \\ & y_{j,j'} \in \{0,1\} \qquad \forall j, \forall j' \neq j \end{aligned}$$

Putting everything together we obtain the following linear integer program:

$$\begin{aligned} \min \quad & \sum_{j=1}^{6} c_j z_j \\ \text{s.t.} \quad & x_j \geq r_j \qquad \forall j \\ & x_j + p_j \leq x_{j'} + M(1 - y_{j,j'}) \qquad \forall j, \forall j' \neq j \\ & x_{j'} + p_{j'} \leq x_j + M y_{j,j'} \qquad \forall j, \forall j' \neq j \\ & x_j + p_j \leq d_j + z_j \qquad \forall j \qquad\qquad x_j \geq 0 \qquad \forall j \\ & z_j \geq 0 \qquad \forall j \\ & y_{j,j'} \in \{0,1\} \qquad \forall j, \forall j' \neq j \end{aligned}$$

□

8.7 Other Models

In the first couple of sections of this chapter, we covered the basics of integer programming models. We can formulate pretty much every problem with that foundation. There are several models that have been widely used in the literature, but we won't spend a lot of time understanding each of them in detail. The main message of this chapter is how to model either/or constraints and implications. The rest are applications, and detecting which model requires one or another way of modeling comes with practice.

Before ending the modeling part of the last chapter of the semester, let's see an unusual use of IP to solve a problem.

Example 8.14 (The world's hardest Sudoku)**.** *Try to solve the following Sudoku, allegedly the world's hardest Sudoku. In case you don't know the rules, here's how to play:*

- *You can only use numbers from 1 to 9 to fill in the blanks*
- *You can use each number exactly once in each row, each column and each 3×3 grid*

8								
		3	6					
	7			9		2		
	5				7			
				4	5	7		
			1				3	
		1					6	8
		8	5				1	
	9					4		

At least I could not solve it by hand, so let's try to use integer programming to solve it. The trick is to use the variables:

$$y_{ijk} = \begin{cases} 1 & \text{if the number in row } i \text{ and column } j \text{ is } k \\ 0 & \text{otherwise} \end{cases} \qquad \forall i, j, k \in \{1, \ldots, 9\}$$

At least for basic Sudoku, we do not have a clear objective function. All we need is a feasible solution under the constraints established above. Then, we can use a constant objective function to make any feasible solution an optimal solution. That is, our objective function can be

$$\min \quad 0$$

The main constraints represent the rules of the game and the numbers that are already placed in the Sudoku. Specifically, we obtain:

- *Exactly one number from 1 to 9 in each row:*

$$\sum_{j=1}^{9} y_{ijk} = 1 \qquad \forall i, k \in \{1, \ldots, 9\}$$

- *Exactly one number from 1 to 9 in each column:*

$$\sum_{i=1}^{9} y_{ijk} = 1 \qquad \forall j, k \in \{1, \ldots, 9\}$$

- *Exactly one number from 1 to 9 in each subgrid. To write these constraints compactly, let's assign a number to each row and column of subgrids, as follows:*

	$c=0$			$c=1$			$c=2$		
$r=0$	8								
			3	6					
		7			9		2		
$r=1$		5				7			
					4	5	7		
				1				3	
$r=2$			1					6	8
			8	5				1	
		9					4		

Then, we obtain the following set of constraints:

$$\sum_{i=3r+1}^{3r+3} \sum_{j=3c+1}^{3c+3} y_{ijk} = 1 \qquad \forall k \in \{1,\dots,9\},\ \forall r,c \in \{0,1,2\}$$

- *We also need to set the numbers that are already part of the Sudoku as constraints. That is, we include the following constraints:*

$$y_{118} = y_{233} = y_{246} = y_{327} = y_{359} = y_{372} = 1 \qquad \textit{(first three rows, i.e., } r=0\textit{)}$$
$$y_{425} = y_{467} = y_{554} = y_{565} = y_{577} = y_{641} = y_{683} = 1 \qquad \textit{(second three rows, i.e., } r=1\textit{)}$$
$$y_{731} = y_{786} = y_{798} = y_{838} = y_{845} = y_{881} = y_{929} = y_{974} = 1 \qquad \textit{(last three rows, i.e., } r=2\textit{)}$$

Hence, putting everything together, we obtain the following model:

$$\begin{aligned}
\min\quad & 0 \\
\textit{s.t.}\quad & \sum_{j=1}^{9} y_{ijk} = 1 \qquad \forall i,k \qquad \textit{(row constraint)} \\
& \sum_{i=1}^{9} y_{ijk} = 1 \qquad \forall j,k \qquad \textit{(column constraint)} \\
& \sum_{i=3r+1}^{3r+3} \sum_{j=3c+1}^{3c+3} y_{ijk} = 1 \qquad \forall k \in \{1,\dots,9\},\ \forall r,c \in \{0,1,2\} \qquad \textit{(subgrid constraints)} \\
& y_{118} = y_{233} = y_{246} = y_{327} = y_{359} = y_{372} = 1 \qquad \textit{(initial solution first three rows, i.e., } r=0\textit{)} \\
& y_{425} = y_{467} = y_{554} = y_{565} = y_{577} = y_{641} = y_{683} = 1 \qquad \textit{(initial solution second three rows, i.e., } r=1\textit{)} \\
& y_{731} = y_{786} = y_{798} = y_{838} = y_{845} = y_{881} = y_{929} = y_{974} = 1 \qquad \textit{(initial solution last three rows, i.e., } r=2\textit{)} \\
& y_{ijk} \in \{0,1\} \qquad \forall i,j,k \in \{1,\dots,9\}
\end{aligned}$$