Data Science and Decision Making: An Elementary Introduction to Modeling and Optimization

David B. Shmoys, Samuel C. Gutekunst, Frans Schalekamp, and D

Jul 25, 2022

CONTENTS

Introduction to the problems and methods of operations research and information engineering focusing on problem areas (including inventory, network design, and resource allocation), the situations in which these problems arise, and several standard solution techniques. In the computational laboratory, students encounter problem simulations and use some standard commercial software packages.

taken from the Spring 2022 Cornell Class Roster

Part I

Introduction

CHAPTER

ONE

WHAT IS OPERATIONS RESEARCH?

It is hard to give a meaningful definition of operations research. For one, it is hard to say what exactly what is operations research (often referred to simply as OR), in contrast to the the emerging areas of data science, or analytics, both of which have been gaining a lot of attention in recent years. For operations research, one such succinct statement might be:

a quantitative approach to decision-making in which a mathematical model of the problem setting is analyzed so as to provide precise guidance for attaining a desired objective.

However, this definition suffers from an attribute that is not unusual — this definition is based on still other terms that themselves would require a definition. For example, what is a "mathematical model"? What makes an approach "quantitative"? What is meant by "desired objective"? How does a mathematical model get "analyzed"?

Furthermore, there is nothing in any of this explanation that provides some understanding of why operations research is a reasonable name for the discipline. One easy answer is: it is not a good name for the discipline. A somewhat more useful answer is: the roots of the field go back to helping guide "operational" decisions. But what is an "operational" decision?

In the summer of 2020, there were a number of operational decisions that needed to be completely rethought because of the COVID-19 pandemic, starting with, is it "safe" to reopen the campus for in-person education? That led to a million-and-one follow-up questions, all seeking to provide guidance to the more general question, how might we most safely "run" the university if we elect to have in-person education. This is still a very general question, but one might ask a more precisely focused one — if we have target limits on the number of COVID-19 cases that might occur at Cornell throughout the semester, and we can test each student every k days, what is the smallest value of k that provides reasonable assurance of not exceeding those targets. Here we again need to ask, what is "reasonable assurance". One element of building a model for this sort of question relies on the mathematical framework of probability, which captures notions of "likelihood". And partnering with such a probabilistic framework are statistical tools: for example, one might model one mode of transmission by the statement — if two people are within 6' of each other for one hour, there is an α percent chance of a person who is positive for COVID-19, infecting the other — but this gives rise to the associated statistical problem, how can you use historical data to estimate α? Indeed, models to answer this type of question, and are still being used to guide the ongoing decisions that inform the university's response to the pandemic, and have been led by Cornell faculty {\it and students} in operations research.

Although probabilistic and statistical models are an extremely important aspects of operations research, this course will focus on so-called deterministic optimization models (i.e., those without a probabilistic element in their set-up); probability and statistics are (mostly) left to be introduced in the subsequent gateway course in operations research, ENGRD 2700.

Let us start with a very simple example of an optimization problem that was, for those students attending this class in person in Fall 2020, literally staring them in the face. Each student attending in person was in an assigned seat that is at least 6' from anyone else. (To be precise, the center of each occupied seat was at least 6' from the outer edge of any other occupied seat.) The seats are in fixed positions. How do we select in which seats to assign students so that the maximum number of students possible can attend a given lecture? This is an operational question, since its answer is one critical element in the functioning of the university. This course will provide a precise language to state this optimization problem, and provide algorithmic and computational tools to solve problems of this sort - in fact, one later lab exercise will be to attack this very application. OR optimization tools were used heavily in redesigning the Cornell fall course roster, which needed to be completely reworked between the time that the decision was made to reopen with in-person classes in early July and August 26th, when enrollment commenced

Hopefully, the examples above have given you a rough idea about some of the kinds of questions that can be addressed with the tools developed by OR, but these are just a very limited set of examples (however important they were to Cornell over the past few months). One of the exciting aspects of OR is that the applications of this discipline come from many, many different areas — health care, environmental preservation, computer design, transportation logistics, financial instruments, genetics, \ldots, the list goes on and on.

A good next step might be to consider one specific example, and study it more in depth, to help understand the rather hard-to-interpret definition of OR given above.

A colleague who was studying for his PhD in astronomy posed following question. For his dissertation, he was studying a particular set of stars. Every few months, he would be allocated one week of access to a powerful radio telescope. The telescope was programmable, and throughout the week, would focus on one particular star for a given length of time (roughly 10 minutes), acquiring data from the signal from that star, and then would be re-positioned to focus on the next star, and so forth. Since it was a radio telescope, this proceeded for a 24-hour cycle (e.g., it could receive the signal 24/7), when the process would begin anew. The time spent re-positioning was, from his perspective, wasted time. It is easy to imagine if the telescope proceeded through the stars in a completely random order, there might be quite a lot of time wasted in re-positioning. In words, the optimization problem (very roughly) is to maximize the amount of time left to observe the stars under investigation.

One natural question is: how were these decisions being made before? The answer here seemed pretty worrisome — there was a "master list" of all of the known stars in the sky, and this induced an ordering of the relatively few stars in the set that were being studied by this colleague. That determined which star would be observed next.

We want to build a mathematical model to guide this decision-making process. The first element of such a model is to understand what the input to such a model might be - what data do we need in order to provide advice for this colleague? One first idea might be to that the desired input is the positional location of the stars being studied. While this is clearly useful, it might not be the entire story. The goal here is to spend as little time as possible re-positioning the telescope; that is, we need to know how the time to re-position the telescope depends on the positions of the stars observed consecutively. So, for example, if we might move from observing Alpha Centuri to next observing Beta Orionis, we need to know the time that would elapse between completing the first observation, and being ready to start the second, which might be denoted

Focus-time[Alpha Centuri, Beta Orionis]

To simplify matters, suppose that there are 100 stars that we wish to observe, and we will call them $1, 2, \ldots, 100$. We denote the set of all 100 stars by $\{1, 2, \ldots, 100\}$ — this is the usual mathematical notation for denoting the set consisting of the integers 1 through 100. In general, we will let n denote the number of stars in our observational set (i.e., $n = 100$). We can use i and j as variables to denote two arbitrary stars in our set — this would be denoted $i \in \{1, 2, \ldots, 100\}$ and $j \in \{1, 2, \ldots, 100\}$, where $i \neq j$. We would then need the data, Focus-time[i,j], for each such pair of stars.

By this point, someone who has studied OR a bit (say, a senior majoring in it at Cornell), but has not seen this telescope observational problem might have the bright idea — "oh, this is just the traveling salesman problem!" And indeed, they would be, more or less, completely right. In words, the traveling salesman problem is often stated as follows: a peddler needs to visit each city exactly once in a given set of cities, starting and ending at home, so as to minimize the total time spent traveling. How should the peddler proceed? In our problem, we have stars, not cities; we have re-positioning times, not travel times, but what we have done is the first conceptual steps in modeling our new application as a traveling salesman problem, which is the next topic in this course.

Part II

The Traveling Salesman Problem

CHAPTER

TWO

THE TRAVELING SALESMAN PROBLEM

The traveling salesman problem is one of the most notorious optimization problems. The setting for the problem is as follows. A salesman starts at his home and has a given set of cities to visit. That is, if his home is in NY, and he must visit Syracuse, Chicago, San Francisco, Los Angeles, Detroit, and Atlanta, then one possible solution is to go from NY to Chicago to San Francisco to Los Angeles to Detroit to Atlanta to Syracuse, and then to return home to NY. Given that he is traveling to many cities and not staying over a Saturday night, he only qualifies for the standard coach airfare between each consecutive pair of cities that he visits. He knows the airfare between each pair of cities. That is, in our example, he might be given the following table of airfares:

	NY	Syracuse	Chicago	SF	LA	Detroit	Atlanta
NY	—	202	135	245	245	169	129
Syracuse	202	—	309	445	445	230	160
Chicago	135	309	—	180	180	105	120
SF	245	445	180	—	39	195	165
LA	245	445	180	39	—	195	165
Detroit	169	230	105	195	195	—	135
Atlanta	129	160	120	165	165	135	—

Attention: Figure 1: A visualization of the traveling salesman problem should be here.

Notice that the data contain a few irregularities (common to the airline industry). The airfare from Syracuse to SF is $445, but the airfare from Syracuse to Atlanta is $129 and the airfare from Atlanta to San Francisco is $165, for a total fare of $294. Unlike in this case, if the table of fares has the property that the cheapest way to go between each pair of cities is to take the non-stop flight, then the fares are said to satisfy the *it triangle inequality*. In cases such as the one above, they are said to *it violate* the triangle inequality. We shall assume that the salesman is pressed for time, and always takes the non-stop flight between each of his stops on his tour. Thus, the cost of the tour proposed above is:

$$135 + 180 + 39 + 195 + 135 + 160 + 202 = 1046.$$

Furthermore, if he were to go from NY to Syracuse to San Francisco to LA to Chicago to Detroit to Atlanta to NY, then the cost would be

$$202 + 445 + 39 + 180 + 105 + 135 + 129 = 1235.$$

Clearly, the first tour is better.

The salesman would like to choose the order in which to visit the cities so as to minimize the total cost of his trip. For the data given above, is the first proposed tour the cheapest one? The traveling salesman problem is an *it optimization problem*. For any optimization problem, there is some notion of what kind of input is expected. For the traveling salesman problem, the input consists of a table of costs, such as the one given above, and it could involve any number of cities. For

any optimization problem, there is also a notion of a *it feasible solution*; that is, a possible answer (though not necessarily the best answer). For this problem, a feasible solution is a tour that visits all of the cities and returns to the starting point. And finally, there is the notion of an *it objection function*, the criterion by which we choose which of the feasible solutions is the best one, the *it optimal solution*. In this case, the objective function is the sum of the costs of flying between each pair of cities that occurs consecutively in the tour. In this case, our objective function is to *it minimize* the value associated with the feasible solution. In other problems, we will be dealing with *it maximization problems*.

CHAPTER

THREE

MATHEMATICAL MODEL

We now wish to formulate a mathematical model of this problem. In order to do this, we must first introduce some notation to describe the input in a concise way. First of all, let the variable n denote the number of cities. So, for the particular input above, $n = 7$. Then, we need some way to describe the costs. We can index our cities as $1, 2, 3, 4, 5, 6, 7$ (sometimes the shorthand $1, 2, \ldots, 7$ is used to indicate this). In other words, NY corresponds to city 1, Syracuse to city 2, and so forth. (If we are describing an arbitrary input, then we refer to cities 1 through n, and this can then be denoted $1, 2, \ldots, n$, even if we have not specified the value that n takes.) We can use a doubly indexed array (or equivalently, a two-dimensional array) to describe the table given above. Let $C[i, j]$ denote the cost of going from city i to city j, where i and j are variables that each can take any integer value between 1 and n. Again, another shorthand notation for this is that the costs are $C[i, j]$, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, n$. One final observation about the input: in our example, the costs had the property that $C[i, j] = C[j, i]$ for each $i = 1, 2, \ldots, n$, and each $j = 1, 2, \ldots, n$. In this case, the input is said to be *it symmetric*. There are cases in which the input might not be symmetric, but this does not change any of the underlying ideas involved.

Now we have given a way to describe the input. How do we describe what is a feasible solution? We can describe the first tour proposed above by saying that city 1 is first, city 3 is second, city 4 is third, and so forth. A more mathematical description of this is to write that $\pi(1) = 1$ (meaning that city 1 is first), $\pi(2) = 3$ (meaning that city 3 is second), $\pi(3) = 4$, $\pi(4) = 5$, $\pi(5) = 6$, $\pi(6) = 7$, and $\pi(7) = 2$. We leave it implicit that as the last part of the tour we return from city $\pi(7)$ to city $\pi(1)$. So the general meaning of $\pi(i) = j$ is that city j is the ith city of the tour. To specify the tour, we must give a value to $\pi(i)$ for each $i = 1, \ldots, n$. (Notice that the 2 has just disappeared from $1, 2, \ldots, n$, and this too is just a further shorthand.) For such a specification of $\pi(i)$, $i = 1, \ldots, n$, to be feasible, what properties must hold? Each city must be visited, and each city must be visited at most once. That is, for each $j = 1, \ldots, n$, there must exist exactly one i (from amongst the integers 1 through n) such that $\pi(i) = j$. In this case, π is called a *it permutation*. Notice that we did not require that the first city in the ordering was the home city for the salesman. There is a good reason for this; a circle does not start or end at any particular point! As long as we have some permutation, we can interpret the cycle as starting at the home city. For example, for the first tour proposed above, we could equally well have set π so that $\pi(1) = 6$, $\pi(2) = 7$, $\pi(3) = 2$, $\pi(4) = 1$, $\pi(5) = 3$, $\pi(6) = 4$, and $\pi(7) = 5$. (Make sure that you understand why this does specify the same tour as the one given above.) Given any permutation π, we can convert it into an equivalent one in which the first city is the home city by starting with the home city and tracing cyclically around the tour π.

Now, how do we give a mathematical description of the objective function? In our example, we simply added $C[\pi(1), \pi(2)]$ and $C[\pi(2), \pi(3)]$ and so forth through $C[\pi(6), \pi(7)]$ and then added the last part (returning home) $C[\pi(7), \pi(1)]$. We need a mathematical shorthand to replace "and so forth". This is done with a summation sign as follows:

$$\sum_{i=1}^{6} C[\pi(i), \pi(i+1)] + C[\pi(7), \pi(1)]$$

and so in general, the objective function is

$$\sum_{i=1}^{n-1} C[\pi(i), \pi(i+1)] + C[\pi(n), \pi(1)].$$

In summary, the traveling salesman problem can now be described as the following mathematically precise computational task.

Traveling Salesman Problem

input:

- an integer n specifying the number of cities, and
- an array $C[i, j]$, for $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, n$ (not necessarily symmetric)

output:

- a permutation π of $1, 2, \ldots, n$

goal: minimize the length of the tour, i.e., minimize

$$\sum_{i=1}^{n-1} C[\pi(i), \pi(i+1)] + C[\pi(n), \pi(1)]$$

Notice that in our way of describing things, the word problem does not refer to one specific input, but rather to a general computational task that, for any input of a certain type, seeks the desired output.

CHAPTER

FOUR

SOLVING A PROBLEM – ALGORITHMS

Let us return to the definition of operations research given in the previous handout. One aspect that we did not touch upon is that we required that the mathematical model be *it analyzed* so it can provide guidance in the given decision-making setting. For an optimization model, the element is thereby an algorithm, a computation procedure that can solve the task at hand. And just as we have described a generic computational task, there should also be a generic procedure that can handle all possible inputs. So, one can think of this as a sufficiently robust software package that can handle any of the inputs you might "reasonably" want to solve.

For the traveling salesman problem, there is a quite simple algorithm that is guaranteed to produce an optimal solution. One might "simply" enumerate all possible permutations, one after the other, evaluate the cost associated with each permutation, and then output the permutation with the cheapest cost. This certainly works correctly for any input. So are we done? Actually, no. The problem is that there are too many permutations for this to be of much practical use. For $n = 100$, how many permutations are there? There are 100 choices for the first city, and for each of those there are 99 choices for the second, and then 98 for the third and so forth. So it seems like there are 100! (100 factorial). In fact, that is a bit overdone, because recall that it doesn't matter which city we start at, so in fact the same tour is counted 100 times. But 99! is big enough. It is more than 10^{155}, more than the number of atoms in our universe.

So the simple algorithm is too slow. There are a few options to consider. One approach, is to develop much more sophisticated algorithms. Throughout this course, you will learn many of the elements used by the best solver for the traveling salesman problem, a package called Concorde (which you can install on your iPhone - you are encouraged to do so). Concorde can easily solve inputs with 1000's or even 10,000 cities, but even that code has its limits.

Another approach is to design simple algorithms that don't necessarily produce an optimal solution, but instead produce good solutions. One natural one is called the *it nearest neighbor rule*: start at one city, and iteratively choose the next city as the one not yet visited that is currently nearest. There are a number of simple algorithms such as this one, and you will be introduced to several during the first lab in the course.

CHAPTER

FIVE

MODELING WITH THE TRAVELING SALESMAN PROBLEM

The traveling salesman problem is an important computational model for a variety of reasons. While the particular application of routing our poor salesman is valid enough, this does not really constitute a real-world application. However, it forms a central component of more complicated models, such as routing a fleet of vehicles for pickups and deliveries, or more concretely, the sort of problem that a UPS depot might solve on a daily basis. It can also be used to model a number of problems that, at first glance, do not seem to be related.

Of course, we have already encountered one such example, the problem of determining the order in which the radio telescope observes the set of stars being studied. In that case, we can model our problem as a traveling salesman problem. The set of stars to be studied corresponds to the set of cities to be visited. The re-positioning times Focus-time[i,j] correspond directly to the costs $C[i, j]$; a feasible solution is again any permutation π of the stars, and the objective is to minimize the total re-positioning time for the telescope:

$$\sum_{i=1}^{n-1} \text{Focus-time}[\pi(i), \pi(i+1)] + \text{Focus-time}[\pi(n), \pi(1)]$$

It is important to note that this exactly captures what is needed in a 24-hour cycle of stellar observations. Hence, we could take our data for re-positioning the telescope, and apply a standard software package for the traveling salesman problem (such as Concorde, which is the state of the art in this case), and be able to interpret the optimal solution for the corresponding traveling salesman input as an (optimal) solution to our telescope problem.

However, while modeling our telescope problem as a traveling salesman problem seems to make sense, and might work on some data sets, it also can fail miserably. Why?

There are, both minor and major reasons why this model does not perfectly represent our computational problem. For minor reasons, one notable issue is that our model of the problem views the telescope positioning problem as a cyclical 24-hour problem. Recall that we had one week of access to the telescope to make observations. So viewing the problem as a cyclical 24-hour problem is nearly correct, if one focuses on the "middle" part of the week; however, we still need to start the week (taking over from how it was used the previous week for another astronomer, and then handing off to yet another person for the following week). However, this is a minor discrepancy and would have only a small effect on the selection of a regular daily pattern.

There is a more fundamental issue. That concerns the statement above: a feasible solution is again any permutation π of the stars. That is not true for our problem. The telescope cannot be positioned to observe a particular star next if it is no longer above the horizon. Thus, to carefully and correctly model the telescope problem, we actually need a more complicated model that takes into account not just "travel times" and "waiting times" (while the telescope is observing a given star), it must also have so-called time-windows for each star, as part of the input, that indicate the period of time (within each 24-hour cycle) that it is observable by our telescope.

Even more generally, this suggests a good rule of thumb - when trying to model a problem in a decision-making setting: start by considering the simplest model! But before attempting to use that model to make actual decisions, it is important to test it, to be sure that the "simplifying assumptions" made along the way are indeed the kind that can be put aside with minimal effect. Hence, the overall design process of thinking about a model might be viewed as follows:

Attention: Figure 2: The cyclical process of formulating and analyzing an OR model should be here.

CHAPTER

SIX

ETCHING VLSI COMPUTER CHIPS

But now we return to the traveling salesman problem itself, and we want to show that sometimes problems that are not quite the same can still be modeled as traveling salesman problem, in that we can use Concorde (or any other general purpose solver for this problem) to find an optimal solution. Here is such an example, which arises in the process of manufacturing VLSI computer chips (VLSI stands for "very large-scale integration", referring to the huge number of transistors that are integrated on a single chip). One step of this process can be viewed in the following simplified way: there is a square (a silicon wafer) on which one is going to etch a sequence of lines (which correspond to electrical connections between different components of the chip). The machine doing the etching first moves to the correct position where the line starts, etches the particular line (as specified by the design) which leaves it at a different point in the square. Then it must move to the correct position for the next line, and so forth. The lines may be etched in any order, but for each line, the etching must proceed in the specified direction (from the given starting point to the given ending point). The etching machine must start and end at the upper left-hand corner of the square (so as not to interfere with the square being put into and out of position for the etching). The VLSI etching optimization problem is to select an order for the lines to be etched so that as little time as possible is taken.

Attention: Figure 3: A visualization of the VLSI etching optimization problem should be here.

We will show that this problem can be viewed as a special case of the traveling salesman problem, or in other words, can be modeled as the traveling salesman problem. More precisely, we will show that given any input to the VLSI problem, we can construct an input for the traveling salesman problem with the property that an optimal solution for this new input (that is, the cheapest tour) can be interpreted as an optimal solution for the VLSI problem. Suppose that N is a variable for the VLSI problem that specifies the number of lines in the input. From any given input to the VLSI problem with N lines, we will construct a TSP input with $n = N + 1$ cities; city 1 corresponds to the upper left-hand corner of this chip (at which the machine starts and ends its etching) and each of cities 2 through $N + 1$ corresponds to one of the N lines in the VLSI input.

The next observation is that in performing the etching of the lines, the machine's time can be divided into two parts: the time actually etching the lines and the time moving the machine between lines when it is not actually etching. No matter what order in which the lines are etched, the time for the first part is the same. So, to minimize the total time for the machine, we should just minimize the total time that we are moving the machine without etching. If cities i and j both correspond to lines (as opposed to the special city 1 which corresponds to the upper left-hand corner) then if the line corresponding to j is etched immediately after the line corresponding i, then we spend the amount of time that it takes to move the machine from the endpoint of i's line to the starting point of j's line (while not etching). Similarly, at the start, we move the machine from the upper left-hand corner to the starting point of the first line etched (while not etching). Finally, we move the machine from the ending point of the last line etched to upper left-hand corner (while not etching). All of the periods in which the machine is not etching are considered in one of these cases. This motivates us to define the cost array as follows: let $C[i, j]$ be the time to move the machine from the ending point of i's line to the starting point of j's line, for each $i = 2, \ldots, N + 1,\ j = 2, \ldots, N + 1$; let $C[1, j]$ be the time to move the machine from the upper left-hand corner to the starting point of j's line, for each $j = 2, \ldots, N+1$; and let $C[i, 1]$ be the time to move the machine from the ending point of i's line to upper left-hand corner, for each $i = 2, \ldots, N + 1$.

As we discussed above, π can be specified so that the tour starts at any particular city, and we shall therefore assume

that $\pi(1)$ is the special city 1. Our explanation above has shown that the total cost of any tour π is exactly the total time that the machine moves while not etching if the lines are etched in the order corresponding to $\pi(2), \pi(3), \ldots, \pi(N+1)$. Consequently, if we find the best tour for the input $C[i,j]$, $i = 1 \ldots, N+1$, $j = 1, \ldots, N+1$, then this yields the ordering of the lines for which the total time to etch the chip is minimized.

This completes the explanation that the VLSI etching problem can be modeled by the traveling salesman problem. Why was this an interesting thing to do? The traveling salesman problem is an extremely well-studied problem. Thousands of man-hours have been invested into devising software packages that solve the traveling salesman problem. By modeling the new problem, the VLSI etching problem, as a traveling salesman problem, we can build off of that experience, and just solve our inputs for the etching problem by using the best software for the traveling salesman problem that we can find. This is one of the main reasons that is important to identify important models in the first place, so that we can then use our experience in solving these models to solve other problems as well.

Part III

Linear Programming and the Simplex Method

CHAPTER

SEVEN

OPTIMIZATION MODELS SO FAR

Let us take a step back, and have a closer look at the range of different optimization models that we have seen so far in this course. All of these formulations had

- decision variables (often denoted by x with some index),
- constraints on these variables, and
- an objective function in terms of these variables.

For instance, given an instance of the Max Flow Problem (a directed graph $G = (V, A)$ with nodes $s, t \in V$ and capacities $u(v, w)$ for every arc $(v, w) \in A$), the Max Flow Problem could be formulated

- with decision variables $f(v, w)$ for each $(v, w) \in A$,
- with two types of constraints:
  - capacity constraints: $0 \leq f(v, w) \leq u(v, w)$ for each $(v, w) \in A$, and
  - flow conservation constraints: $\sum_{(w,v)\in A} f(w, v) = \sum_{(v,w)\in A} f(v, w)$ for each node $v \in N \setminus \{s, t\}$,
- and the objective (the value of the flow) expressed in terms of the variables: $\sum_{(w,t)\in A} f(w, t) - \sum_{(t,w)\in A} f(t, w)$.

In short, we had the following mathematical programming formulation for the Max Flow Problem.

Max Flow Problem

$$\begin{aligned} \text{maximize} \quad & \sum_{(w,t)\in A} f(w,t) - \sum_{(t,w)\in A} f(t,w) & \\ \text{st} \quad & \sum_{(w,v)\in A} f(w,v) = \sum_{(v,w)\in A} f(v,w) & \text{for each node } v \in N \setminus \{s,t\}, \\ & 0 \leq f(v,w) \leq u(v,w) & \text{for each } (v,w) \in A. \end{aligned}$$

As a second example, recall the Assignment Problem. Here we had decision variables x_{ij} for each worker i and task j, taking on a value 0 or 1. These had the following interpretation: if $x_{ij} = 1$ then worker i is assigned to task j, and if $x_{ij} = 0$ then worker i is *not* assigned to task j. Using these decision variables, we could then express the constraint that every task must be assigned to exactly one worker as

$$\sum_{i=1}^{n} x_{ij} = 1 \text{ for each task } j = 1, \ldots, n,$$

and the constraint that every worker must be assigned exactly one task as

$$\sum_{j=1}^{n} x_{ij} = 1 \text{ for each worker } i = 1, \ldots, n.$$

The objective for this problem is the total time required to execute all tasks, which can be expressed in terms of the decision variables as $\sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij} x_{ij}$.

Summarizing, we had the following mathematical programming formulation for the Assignment Problem.

Assignment Problem

$$\begin{aligned} \text{minimize } & \sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij} x_{ij} & \\ \text{st } & \sum_{i=1}^{n} x_{ij} = 1 & \text{for each task } j = 1, \ldots, n, \\ & \sum_{j=1}^{n} x_{ij} = 1 & \text{for each worker } i = 1, \ldots, n, \\ & x_{ij} \in \{0, 1\} & \text{for each } i = 1, \ldots, n;\ j = 1, \ldots, n. \end{aligned}$$

Ignoring the fact that the decision variables in the Assignment Problem can only take on two values (0 or 1), we see that the constraints and objective functions of both these optimization formulations involve only *linear functions* in the decisions variables!

Definition

A function $f(x_1, x_2, \ldots, x_n)$ is **a linear function** (in the variables $x_1, x_2, \ldots, x_n$) if it can be written as

$$f(x_1, x_2, \ldots, x_n) = c_0 + c_1 x_1 + c_2 x_2 + \cdots + c_n x_n$$

for some *constants* $c_0, c_1, \ldots, c_n$ (i.e., numbers that do not depend on $x_1, x_2, \ldots, x_n$).

In fact, all the problems that we have studied so far in this course have formulations that only involve linear functions, and lots of other problems can be modeled using only linear functions as well! In this handout we will study how to find optimal solutions to such mathematical programs.

CHAPTER

EIGHT

LINEAR PROGRAMMING

Definition

A **Linear Program** (or Linear Programming Problem) is an optimization problem with a {\em linear} objective function, and constraints that are either "$\leq$", "$\geq$" or "$=$"-constraints, that have *linear* functions on *both* sides of the (in)equality.

Examples. Examples of Linear Programs (LPs for short) are the mathematical programming formulation of the Max Flow Problem above, as well as the mathematical programming formulation of the Assignment Problem, if we change the constraints $x_{ij} \in \{0, 1\}$ for all i, j to $0 \leq x_{ij} \leq 1$ for all i, j. (We will get back to mathematical programs with restrictions that variables have to be integer in the very near future.)

$$\begin{aligned} \text{minimize } & 2x \\ \text{such that } & x \geq 5 \\ & 2x \leq 16 \\ & 5x \leq 50. \end{aligned}$$

As you know, as soon as we define a new problem in this course (in this case a generalization of the problems we saw before), our goals then are (1) to come up with an algorithm to find an optimal solution for this problem, and (2) to find a (relatively easy) argument to show that a solution is indeed optimal. We will now tackle the first goal. The second goal will also be addressed here, but we will actually see a more satisfying argument at a later point in this course.

CHAPTER

NINE

FINDING AN OPTIMAL SOLUTION OF A LINEAR PROGRAM

Let us consider the simple example in the previous paragraph.

$$\begin{aligned} &\text{minimize } 2x \\ &\text{such that } x \geq 5 \\ &\qquad 2x \leq 16 \\ &\qquad 5x \leq 50. \end{aligned}$$

We can actually solve this problem by inspection: an optimal solution is $x^* = 5$, with objective value $2x^* = 10$. How did we see this, and can we generalize this intuition to linear programs that look more complicated? For this example, we can simplify the constraints to $x \geq 5$, $x \leq 8$ and $x \leq 10$, and combine these to be $x \in [5, 8]$. To minimize the function $2x$ we obviously want to make x as small as possible, i.e., 5. It is unclear how this reasoning generalizes to give us a method for a linear program with more than one variable, however.

Here is another way of reasoning: graphically.

Attention: 1D graphical solve should be here

We associate values of x with points on the number line. The border of the region of values of x that make a constraint feasible is drawn for every constraint, with a pair of arrows indicating on which side the values of x lie that make the constraint satisfied. The shaded region then is the intersection of all these regions: it is the region containing the values of x for which all constraints are satisfied. To minimize $2x$ we find the point in the shaded region that makes this function as small as possible: $x^* = 5$.

Let's see if this idea generalizes to an LP with two variables. Let's consider the following linear program.

$$\begin{aligned} &\text{maximize} 3x + 2y \\ &\text{such that } x \leq 4 \\ &\qquad y \leq 2 \\ &\qquad 2x + y \leq 6 \\ &\qquad y - x \leq 1 \\ &\qquad x \geq 0 \\ &\qquad y \geq 0. \end{aligned}$$

We can associate values of the variables with a point on the plane, associating the first coordinate with the x-variable, and the second coordinate with the y-variable. Again, we plot the constraints as regions in the plane that correspond to values of the variables such that a constraint is satisfied.

Attention: 2D feasible region should be here

We now know which values for x and y are {\em feasible}, by which we mean values so that all constraints are satisfied. We will refer to the region of points corresponding to feasible solutions as the {\em feasible region} (the shaded region in both pictures above).

Let's find an optimal solution: in the drawing this corresponds to a point in the feasible region that has the best objective value. We can do this graphically as well. Let's draw a line through all points corresponding to solutions with objective value 6 (where 6 is just a arbitrary value that I chose here). This is the line $3x + 2y = 6$, and we add it to the figure (it is the dashed line in the drawing below).

Attention: Isoprofit line (6) should be here

The line $3x + 2y = 6$ intersects the feasible region, which means all points in this intersection of the line and the feasible region correspond to {\em feasible} solutions that have objective value 6. Are there solution with higher objective value? Let's draw the line $3x + 2y = 9$.

Attention: Isoprofit line (9) should be here

The line $3x + 2y = 9$ also intersects the feasible region, which means all points in this intersection of this line and the feasible region correspond to feasible solution with objective value 9. How about solutions with objective value 12? We add the line $3x + 2y = 12$.

Attention: Isoprofit line (12) should be here

This line has an empty intersection with the feasible region, from which we can conclude that there are no feasible solutions with objective value 12. But what is now the best objective value we can achieve? From the picture above we clearly get the right idea: we can think of the objective value as defining parallel lines, one for each possible objective value. We are trying to push this line as far to the top right as possible, while it still intersects the feasible region.

Attention: Isoprofit line (10) should be here

How did can we tell that 10 is going to be the optimal objective value of the Linear Program that we were trying to solve? We can conclude from the earlier pictures that the line representing solutions with the highest objective value are only going to intersect the feasible region in one point: the point that is the intersection of the lines that represent the constraints $y \leq 2$ and $2x + y \leq 6$ (i.e., the lines $y = 2$ and $2x + y = 6$). Simple algebra immediately gives the intersection point, namely $x^* = 2$, $y^* = 2$, which corresponds to an objective value of $3x^* + 2y^* = 10$.

CHAPTER

TEN

SOLDIERING ON - LINEAR PROGRAMS WITH THREE VARIABLES

How about finding the optimal solution to the following LP:

$$\begin{aligned} \text{maximize } & 2x_1 + x_2 + x_3 \\ \text{st } & x_1 \leq 4 \\ & x_2 \leq 4 \\ & x_1 + x_2 \leq 6 \\ & -x_1 + 2x_3 \leq 4 \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

With the help of a computer (or exercising an inordinate amount of patience) one might be able to plot this in 3-dimensional space. In fact, the accompanying lab exercise will give you a computational tool to do exactly this, but without such a tool, carefully graphing this problem would not really be feasible. And even after solving this LPs with 3 variables, we would not be content, because we would also want to solve LPs with 4, 5, 10, 27, etc. variables. We thus need a different method, and abandon our current effort of solving LPs graphically. We can console ourselves that the previous hard labor at least gave us some geometric insight in what the feasible region and the objective function of a Linear Program look like (and what points to consider when looking for an optimal solution).[1]

[1] In the first example the constraints corresponded to half-open intervals of form $[a, \infty)$ or $(-\infty, b]$, the border of this region being a (0-dimensional) point. In the second, the constraints corresponded to *half-planes*, planar regions consisting of all points on one side of a line, the border being a (1-dimensional) line. In 3 dimensions the constraints will correspond to *3-dimensional half-spaces*, 3-dimensional regions consisting of all points on one side of a plane, the border being a (2-dimensional) plane. I think you can see a pattern here. In n-dimensional space the constraints will correspond to n-dimensional half-spaces, n-dimensional regions consisting of all points on one side of a $(n-1)$-dimensional half-space, the border being an $(n-1)$-dimensional half-space. Good luck picturing that!

CHAPTER

ELEVEN

SIMPLEX METHOD

How about finding the optimal solution to the following LP:

$$\begin{aligned} \text{maximize } & 14 - \tfrac{3}{2}x_4 - x_6 - \tfrac{1}{2}x_7 \\ \text{st } & x_4 \leq 4 \\ & x_4 - x_6 \leq 2 \\ & -x_4 + x_6 \leq 2 \\ & \tfrac{1}{2}x_4 + \tfrac{1}{2}x_7 \leq 4 \\ & x_4, x_6, x_7 \geq 0. \end{aligned}$$

This is actually easy! (Find it before reading on.)

The coefficients of all variables in the objective function are negative, and all variables are constrained to be nonnegative. This means that we have an *upper bound* on the objective value of any feasible solution of 14. Now consider the solution $x_4^* = x_6^* = x_7^* = 0$. This is a feasible solution (all constraints are satisfied), and has objective value equal to 14, matching the upper bound! This means there is no feasible solution with a better objective value, which means we found an optimal solution!

The method that we now present transforms LPs (of a certain type) into such an easy LP, for which we can immediately read off an optimal solution, because the coefficients of all variables in the objective function will be negative, and all variables are constrained to be nonnegative. We will just use simple algebra to rewrite the LP, *while not changing the feasible region or the objective function*.

The type of LPs that we will tackle here are LPs that are maximization problems, having only "$\leq$"-constraints with nonnegative constants as right-hand sides, and nonnegative variables. It may seem that we have narrowed the type of LP that we are considering drastically, but all of these restrictions can be dropped. You will learn more about this in your homework, and when you take ORIE 3300.

11.1 First Steps

Recall the LP we are trying to solve:

$$\begin{aligned} \text{maximize } & 2x_1 + x_2 + x_3 \\ \text{such that } & x_1 \leq 4 \\ & x_2 \leq 4 \\ & x_1 + x_2 \leq 6 \\ & -x_1 + 2x_3 \leq 4 \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

The first thing we do is to rewrite each constraint in the form $0 \leq \cdots$. This may seem silly, but there is a method to the madness that will soon become apparent.

$$\begin{aligned} \text{maximize} \quad & 2x_1 + x_2 + x_3 \\ \text{such that } 0 &\leq 4 - x_1 \\ 0 &\leq 4 - x_2 \\ 0 &\leq 6 - x_1 - x_2 \\ 0 &\leq 4 + x_1 - 2x_3 \end{aligned}$$

$$x_1, x_2, x_3 \geq 0.$$

Recall that you know how to use algebra to solve systems of equations. So let's make the LP more resemble a system of equations, by introducing new variables, that we constrain to be nonnegative just like the other variables.

$$\begin{aligned} \text{maximize } z &= 2x_1 + x_2 + x_3 \\ \text{such that } x_4 &= 4 - x_1 \\ x_5 &= 4 - x_2 \\ x_6 &= 6 - x_1 - x_2 \\ x_7 &= 4 + x_1 - 2x_3 \end{aligned}$$

$$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0.$$

Note that we have only changed the appearance of the LP! The feasible region of the original LP and the LP here in terms of x_1, x_2 and x_3 are exactly the same. And we have added the variable z so that we can handle the objective function in a manner exactly analogously to how we handle the constraints.

We will call this our first *dictionary*, by which we mean the following specific form of writing the LP using equations.

Definition

We say an LP is in **dictionary form** if all constraints are equality constraints, and all variables are constrained to be nonnegative. Further, each variable only appears on the left-hand side *or* the right-hand side of the constraints (not both), and each constraint has a *unique variable* on the left-hand side. The objective function is in terms of the variables that appear on the right-hand side of the constraints only. Finally, all constants on the right-hand side of the constraints are nonnegative.

One nice consequence of this definition is that, similar to the LP where we immediately saw an optimal solution, we can also read off a *feasible* solution here: we can set all variables that appear on the right hand sides (x_1, x_2 and x_3) to 0, and a feasible solution would then have $x_4 = 4$, $x_5 = 4$, $x_6 = 6$ and $x_7 = 4$. We call the feasible solution that we get from a dictionary by setting all variables that appear on the right hand sides of the constraints equal to 0, *the feasible solution associated with this dictionary*.

The objective value of the feasible solution associated with the dictionary above is 0. Another nice consequence of writing the LP in dictionary form is that we see that there may be better solutions, because variables x_1, x_2 and x_3 all appear with a *positive* coefficient in the objective function, and all these variables are equal to 0 in this solution.

11.2 Rewriting the LP

This motivates us to do the following. We want to find a solution where x_1, x_2 or x_3 is positive — let's choose x_1. We will find a different dictionary for the LP, so that x_1 will now appear *on the left-hand side* of one of the constraints.

Let's think about increasing x_1 as much as possible, while maintaining a feasible solution. The first constraint ($x_4 = 4 - x_1$) tells us that x_1 cannot be increased to more than 4, because otherwise x_4 has to become negative, and this is not feasible. The second constraint ($x_5 = 2 - x_2$) does not put any limit on how much x_1 can increase, because increasing x_1 does not affect either side of this equality. The third constraint ($x_6 = 6 - x_1 - x_2$) puts a limit of 6 on the value of x_1, before x_6 has to become negative. Finally, the fourth constraint ($x_7 = 4 + x_1 - 2x_3$) does not put any restrictions on the increase of x_1 — when x_1 increases, x_7 will increase as well, and this is not a problem for feasibility.

We found a limit on how much we can increase x_1 (namely 4), and the constraint that put this limit on the amount that we can increase x_1 (the first constraint). Note that if we want to increase x_1 to 4, that x_4 (the left-hand side of the first constraint) will have to become equal to zero. Because of our convention about variables on the right- and left-hand side of the constraints, this means that we could actually move x_4 from the left-hand side to the right-hand side, which is good, because we have to move x_1 from the right-hand side to the left-hand side if it is non-zero!

We rewrite the first equation, by solving for x_1, obtaining $x_1 = 4 - x_4$. Note that this equation is equivalent to $x_4 = 4 - x_1$. By replacing the first equation with this equation, we have exactly the same feasible region for the LP that we are trying to solve.

If we just replace the first equation, we no longer have an LP in dictionary form however, because x_1 still appears on the right-hand side of some of the other equations. This is easily fixed: we can substitute $4 - x_4$ for every occurrence of x_1 on the right-hand side of the equations (because we just found that $x_1 = 4 - x_4$). x_4 will be a right-hand side variable, so it is no problem that it will appear on the right-hand side of other equations as well. (In general, there may be other right-hand side variables in the expression for the variable we are increasing, but never left-hand side variables!)

After that long discussion, we are finally ready to rewrite the LP for the first time:

$$\begin{aligned}
\text{maximize } z &= 2(4 - x_4) + x_2 + x_3 \\
\text{such that } x_1 &= 4 - x_4 \\
x_5 &= 4 - x_2 \\
x_6 &= 6 - (4 - x_4) - x_2 \\
x_7 &= 4 + (4 - x_4) - 2x_3
\end{aligned}$$

$$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0.$$

Note that we also substituted $4 - x_4$ for x_1 in the objective function. Simplifying gives:

$$\begin{aligned}
\text{maximize } z &= 8 - 2x_4 + x_2 + x_3 \\
\text{such that } x_1 &= 4 - x_4 \\
x_5 &= 4 - x_2 \\
x_6 &= 2 + x_4 - x_2 \\
x_7 &= 8 - x_4 - 2x_3
\end{aligned}$$

$$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0.$$

This is our second dictionary. Notice what we accomplished here: first of all, this LP is equivalent to the LP we started with (we only did some algebraic manipulations); second of all, it is in a form that we want our LPs to be in (it is in dictionary form: every variable appears only on the right-hand side of the equations, or only on the left-hand side of one equation, and we can associate a feasible solution with this particular form of the LP, by setting all variables on the right-hand side equal to zero); and finally the objective value of the solution that we associate with this form is higher than the objective value of the solution that we had before (we can actually immediately see the objective value z is 8 because only right-hand side variables appear in the objective function). The solution associated with this dictionary is $x_2 = x_3 = x_4 = 0$, and $x_1 = 4$, $x_5 = 4$, $x_6 = 2$, $x_7 = 8$.

11.3 Rewriting the LP Again

Can we find a feasible solution with an even higher objective value? Well, x_2 and x_3 appear in the objective function with a positive coefficient, and these variables are currently 0 in the solution that we associate with the current dictionary. So let's try and see if we can do the same thing again, now for x_2.

Limit on increase of x_2. The first constraint does not put a limit on the increase of x_2, the second limits the increase to 4, before x_5 becomes negative, the third limits the increase to 2, before x_6 becomes negative, and the fourth does not put a limit on the increase of x_2. We want x_2 to increase as much as possible, but we cannot increase it by more than 2 because of the third constraint.

Substitute x_2 We solve the third equation for x_2, and replace the third equation with the expression we obtain: $x_2 = 2 + x_4 - x_6$. We substitute $2 + x_4 - x_6$ for every other appearance of x_2 in the LP,

$$\begin{aligned}
\text{maximize } z &= 8 - 2x_4 + (2 + x_4 - x_6) + x_3 \\
\text{such that } x_1 &= 4 - x_4 \\
x_5 &= 4 - (2 + x_4 - x_6) \\
x_2 &= 2 + x_4 - x_6 \\
x_7 &= 8 - x_4 - 2x_3
\end{aligned}$$

$$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0,$$

and simplify:

$$\begin{aligned}
\text{maximize } z &= 10 - x_4 - x_6 + x_3 \\
\text{such that } x_1 &= 4 - x_4 \\
x_5 &= 2 - x_4 + x_6 \\
x_2 &= 2 + x_4 - x_6 \\
x_7 &= 8 - x_4 - 2x_3
\end{aligned}$$

$$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0,$$

We have found the feasible solution $x_3 = x_4 = x_6 = 0$, and $x_1 = 4$, $x_2 = 2$, $x_5 = 2$, $x_7 = 8$, with objective value 10.

11.4 And Again

The variable x_3 appears in the objective function with a positive coefficient, and this variable is currently 0 in the solution that we associate with the current dictionary. So let's see if we can further improve the solution, now by increasing x_3.

Limit on increase of x_3. The first constraint does not put a limit on the increase of x_3, neither does the second or the third. The fourth is the only constraint that puts a limit on the increase of x_3.

Substitute x_3. We replace the fourth equation with $x_3 = 4 - \frac{1}{2}x_4 - \frac{1}{2}x_7$, and substitute $4 - \frac{1}{2}x_4 - \frac{1}{2}x_7$ for every other appearance of x_3 in the LP,

$$\begin{aligned}
\text{maximize } z &= 10 - x_4 - x_6 + (4 - \tfrac{1}{2}x_4 - \tfrac{1}{2}x_7) \\
\text{such that } x_1 &= 4 - x_4 \\
x_5 &= 2 - x_4 + x_6 \\
x_2 &= 2 + x_4 - x_6 \\
x_3 &= 4 - \tfrac{1}{2}x_4 - \tfrac{1}{2}x_7
\end{aligned}$$

$$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0,$$

and simplify:

$$\begin{aligned}
\text{maximize } z &= 14 - \tfrac{3}{2}x_4 - x_6 - \tfrac{1}{2}x_7 \\
\text{such that } x_1 &= 4 - x_4 \\
x_5 &= 2 - x_4 + x_6 \\
x_2 &= 2 + x_4 - x_6 \\
x_3 &= 4 - \tfrac{1}{2}x_4 - \tfrac{1}{2}x_7
\end{aligned}$$

$$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0,$$

11.5 Done!

Note that this last LP is exactly in the form that we wanted (many pages ago): The coefficients of all variables in the objective function are negative, and all variables are constrained to be nonnegative. This means that we have an *upper bound* on the objective value of any feasible solution of 14. We also have a feasible solution with objective value 14: $x_4^* = x_6^* = x_7^* = 0$ and $x_1^* = 4, x_5^* = 2, x_2^* = 2, x_3^* = 4$. The objective value matches the upper bound, which means there is no feasible solution with a better objective value, which means we found an optimal solution!

And let's do it a quick check and plug this solution into the LP we started with:

$$\begin{aligned}
\text{maximize } & 2x_1 + x_2 + x_3 \\
\text{such that } & x_1 \leq 4 \\
& x_2 \leq 4 \\
& x_1 + x_2 \leq 6 \\
& -x_1 + 2x_3 \leq 4 \\
& x_1, x_2, x_3 \geq 0.
\end{aligned}$$

Note that $x_1^* = 4, x_2^* = 2, x_3^* = 4$ is indeed feasible (all constraints are satisfied), and indeed has objective value 14 as expected.

CHAPTER

TWELVE

SUMMARY OF SIMPLEX METHOD

We described above, by example, a method for solving LPs, known as the *Simplex Method*. It can be summarized as follows.

Simplex Method

Initialization: Set up the first dictionary: replace the inequality constraints by equality constraints by introducing new nonnegative variables. The first dictionary has the new variables on the left hand side, and the other variables on the right hand side.

Loop: Then repeat the following:

- identify a variable that has positive coefficient in objective function
- determine how much this variable can increase, and the most restrictive constraint
- rearrange dictionary, by solving for the increasing variable in the most restrictive constraint, and substituting this in the other equalities and the objective. ```

CHAPTER

THIRTEEN

ARGUMENT THAT SOLUTION IS OPTIMAL

The method consists of rewriting the constraints using algebra. By rewriting the equalities, the feasible region is not changed, and neither is the objective function. So the new LP is exactly the same as the old LP, just written differently. The final LP gives both an upper bound on the objective value, and a feasible solution with the objective value equal to this bound - this must be the optimal solution to this last LP, and because the LP is equivalent to the LP we started with, it must also be an optimal solution to that LP.

CHAPTER

FOURTEEN

GEOMETRY OF THE SIMPLEX METHOD

Let's revisit the 2-dimensional example earlier, to visualize what the Simplex Method is doing. The LP we considered was

$$\begin{aligned} \text{maximize } & 3x + 2y \\ \text{such that } & x \leq 4 \\ & y \leq 2 \\ & 2x + y \leq 6 \\ & y - x \leq 1 \\ & x \geq 0 \\ & y \geq 0. \end{aligned}$$

The feasible region looks as follows.

Attention: Feasible region should be here.

The first step of the Simplex Method rewrites the LP by introducing a new variable for every constraint, let's call them z_1, z_2, z_3 and z_4, and a new variable z_0 for the objective function.

$$\begin{aligned} \text{maximize } z_0 &= \quad 3x+2y \\ \text{such that } z_1 &= 4- \ x \\ z_2 &= 2 \quad - \ y \\ z_3 &= 6-2x- \ y \\ z_4 &= 1+ \ x- \ y \end{aligned}$$

$$x, y, z_1, z_2, z_3, z_4 \geq 0.$$

The feasible solution associated with this dictionary is $x = y = 0$, and $z_1 = 4$, $z_2 = 2$, $z_3 = 6$, $z_4 = 1$. This corresponds to the origin in our drawing (we will ignore the variables z_1, z_2, z_3 and z_4, because we are only interested in x and y originally, and we know that the feasible region of this first dictionary is the same as the feasible region of the original LP in terms of the original variables).

Attention: Initial feasible solution.

We can now choose to increase either x or y - let's choose y. The most restricting constraint is constraint number 4. We solve for the equation for y, and get $y = 1 + x - z_4$. We replace the fourth constraint with this equation, and substitute

$1 + x - z_4$ for y in every other constraint and the objective function, and get

$$\begin{aligned} \text{maximize } z_0 = \quad & 3x+2(1+x-z_4) \\ \text{such that } z_1 = 4- \ & x \\ z_2 = 2 \quad - \ & (1+x-z_4) \\ z_3 = 6-2x- \ & (1+x-z_4) \\ y = 1+ \ & x- \quad z_4 \end{aligned}$$

$$x, y, z_1, z_2, z_3, z_4 \geq 0.$$

which simplifies to

$$\begin{aligned} \text{maximize } z_0 = 2+5x-2z_4 \\ \text{such that } z_1 = 4- \ x \\ z_2 = 1- \ x+ \ z_4 \\ z_3 = 5-3x+ \ z_4 \\ y = 1+ \ x- \ z_4 \end{aligned}$$

$$x, y, z_1, z_2, z_3, z_4 \geq 0.$$

The feasible solution associated with this dictionary is $x = z_4 = 0$, and $y = 1$, $z_1 = 4$, $z_2 = 1$, $z_3 = 5$. This corresponds to the point $(0, 1)$ in our drawing (again ignoring the variables z_1, z_2, z_3 and z_4, because we are only interested in x and y).

Attention: First iteration should be here

The Simplex Method increased the y-variable as much as possible, until it hit one of the constraints, in particular the constraint $y - x \leq 1$ here! In the dictionary this corresponded to the constraint $z_4 = 1 + x - y$, which was indeed the most limiting constraint.

We can now increase x. The most restricting constraint is constraint number 2. We solve for the equation for x, and get $x = 1 - z_2 + z_4$. We replace the second constraint with this equation, and substitute $1 - z_2 + z_4$ for x in every other constraint and the objective function, and get

$$\begin{aligned} \text{maximize } z_0 = 2+5(1-z_2+z_4)-2z_4 \\ \text{such that } z_1 = 4- \ (1-z_2+z_4) \\ x = 1- \quad z_2+ \ z_4 \\ z_3 = 5-3(1-z_2+z_4)+ \ z_4 \\ y = 1+ \ (1-z_2+z_4)- \ z_4 \end{aligned}$$

$$x, y, z_1, z_2, z_3, z_4 \geq 0.$$

which simplifies to

$$\begin{aligned} \text{maximize } z_0 = 7-5z_2+3z_4 \\ \text{such that } z_1 = 3+ \ z_2- \ z_4 \\ x = 1- \ z_2+ \ z_4 \\ z_3 = 2+3z_2-2z_4 \\ y = 2- \ z_2 \end{aligned}$$

$$x, y, z_1, z_2, z_3, z_4 \geq 0.$$

The feasible solution associated with this dictionary is $z_2 = z_4 = 0$, and $x = 1$, $y = 2$, $z_1 = 3$, $z_3 = 2$. This corresponds to the point $(1, 2)$ in our drawing.

Attention: Second iteration should be here

The Simplex Method now increased the x-variable as much as possible, until it hit one of the constraints, in particular the constraint $y \leq 2$ here — this constraint actually limited the increase of x, because y is set equal to $1 + x - z_4 = 1 + x$ (because we are keeping z_4 fixed at 0)! This implies the limit $1 + x \leq 2$, or $x \leq 1$, which is indeed the limit that the Simplex Method put on x. In the dictionary this corresponded to the constraint $z_2 = 1 - x + z_4$, which was indeed the most limiting constraint.

We can start to see a pattern here.

Note: The point is that the solution associated with the dictionaries in Simplex Method correspond to "corner points" of the feasible region. Intuitively we can interpret every step in the Simplex Method as moving the current feasible solution to a next one, by moving along a boundary line of the feasible region, until a next constraint is hit.

Let's finish this example for completeness' sake - fully understanding the reasoning about the relationship of the constraints in the dictionary and the original LP is not as important as the geometric insight that we have gained.

Increasing z_4 can still improve the solution. The most restricting constraint is the third constraint. We solve for the equation for z_4, and get $z_4 = 1 + \frac{3}{2}z_2 - \frac{1}{2}z_3$. We replace the third constraint with this equation, and substitute $1 + \frac{3}{2}z_2 - \frac{1}{2}z_3$ for z_4 in every other constraint and the objective function, and get

$$\begin{aligned}
\text{maximize } z_0 &= 7 - 5z_2 + 3(1 + \tfrac{3}{2}z_2 - \tfrac{1}{2}z_3) \\
\text{such that } z_1 &= 5 + z_2 - (1 + \tfrac{3}{2}z_2 - \tfrac{1}{2}z_3) \\
x &= 1 - z_2 + (1 + \tfrac{3}{2}z_2 - \tfrac{1}{2}z_3) \\
z_4 &= 1 + \tfrac{3}{2}z_2 - \tfrac{1}{2}z_3 \\
y &= 2 - z_2
\end{aligned}$$

$$x, y, z_1, z_2, z_3, z_4 \geq 0.$$

which simplifies to

$$\begin{aligned}
\text{maximize } z_0 &= 10 - \tfrac{1}{2}z_2 - \tfrac{3}{2}z_3 \\
\text{such that } z_1 &= 4 - \tfrac{1}{2}z_2 + \tfrac{1}{2}z_3 \\
x &= 2 + \tfrac{1}{2}z_2 - \tfrac{1}{2}z_3 \\
z_4 &= 1 + \tfrac{3}{2}z_2 - \tfrac{1}{2}z_3 \\
y &= 2 - z_2
\end{aligned}$$

$$x, y, z_1, z_2, z_3, z_4 \geq 0.$$

The feasible solution associated with this dictionary is $z_2 = z_3 = 0$, and $x = 2$, $y = 2$, $z_1 = 4$, $z_4 = 1$. This corresponds to the point $(2, 2)$ in our drawing.

Attention: Third iteration should be here

The Simplex Method now increased the z_4-variable as much as possible, until it hit one of the constraints, in particular the constraint $2x + y \leq 6$ here - this constraint actually limited the increase of z_4, because x is set equal to $1 - z_2 + z_4 = 1 + z_4$ (because we are keeping z_2 equal to 0) and y is set equal to $2 - z_2 = 2$. Note that $6 \geq 2x + y = 2(1 + z_4) + 2 = 4 + 2z_4$ which implies the limit $z_4 \leq 1$. In the current dictionary $2x + y \leq 6$ corresponded to the constraint $z_3 = 2 + 3z_2 - 2z_4$, which was indeed the most limiting constraint. Also note that this indeed means that x now increases to 2 (because $x = 1 + z_4$).