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.