ORIE 5380, CS 5727: Optimization Methods

Huseyin Topaloglu
ht88@cornell.edu
School of Operations Research and Information Engineering
Cornell Tech

Contents

Formulating a Linear Program and Excel's Solver

In this chapter, we use examples to understand how we can formulate linear programs to model decision-making problems and how we can use Microsoft Excel's solver to obtain the optimal solution to these linear programs.

1 Allocating Servers Between Two Customer Types

Assume that we have 1000 servers to lease to users on a daily basis. There are two types of users that we serve, standard users and power users. Standard users pay \$3 per day for each server and consume 1 unit of energy per day for each server they use. Power users pay \$4 per day for each server and consume 2 units of energy per day for each server they use. We have 1600 units of energy available per day. We are interested in figuring out how many servers to lease to standard and power users to maximize the revenue per day.

To formulate this problem as a linear program, we need to identify the decision variables and express the objective function and the constraints as a function of the decision variables. The decision variables are the quantities whose values we want to determine to attain our objective. Our objective is to maximize the revenue per day. To attain this objective, we need to determine the numbers of servers that we lease to standard and power users. Thus, the decision variables for this problem are as follows.

After we identify the decision variables, we need to express the objective function as a function of the decision variables. In this problem, our objective is to maximize the revenue per day. We obtain a revenue of \$3 for each standard user that we serve and we obtain a revenue of \$4 for each power user that we serve. As a function of the decision variables above, we can express the revenue per day as $3\,x_s + 4\,x_p$, which is our objective.

Next, we need to express the constraints as a function of the decision variables. The number of available servers and the amount of energy available per day restrict our decisions. As a function of our decision variables, the total number of servers that we lease is given by $x_s + x_p$ and the number of servers that we lease cannot exceed 1000. Thus, one constraint we have is $x_s + x_p \leq 1000$. Also, we consume 1 unit of energy per day for each server that we leave to a standard user and 2 units of energy per day for each server that we lease to a power user. So, as a function of our decision variables, the total energy consumption per day is $x_s + 2\,x_p$ and the total energy consumption per day cannot exceed 1600 units. Thus, another constraint we have is $x_s + 2\,x_p \leq 1600$. Note that both of our constraints are expressed with a less than or equal to sign, but we can express constraints

with a greater than or equal to sign or with an equal to sign. Which constraint type we use depends on the problem statement. Finally, the number of servers that we lease to each type of users cannot be negative. Therefore, we have the constraints $x_s \geq 0$ and $x_p \geq 0$.

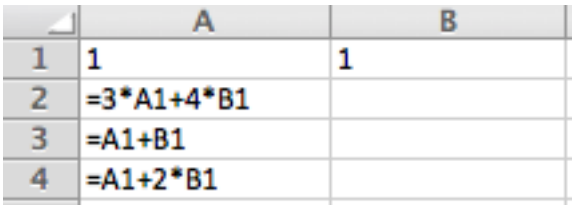
Putting the discussion above together, the optimization problem that we want to solve can be expressed as

The set of equations above characterize an optimization problem. The first row shows the objective function and max emphasizes the fact that we are maximizing our objective function. The second, third and fourth rows show the constraints. The acronym st stands for subject to and it emphasizes that we are maximizing the objective function subject to the constraints in the second, third and fourth rows. Since the objective function and the constraints are linear functions of the decision variables, the optimization problem characterized by the set of equations above is called a linear program. We study linear programs for a significant portion of this course, but there are optimization problems whose objective functions and constraints are not necessarily linear functions of the decision variables. Such optimization problems are called nonlinear programs.

A pair of values of the decision variables (x_s, x_p) that satisfies all of the constraints in the linear program above is called a feasible solution to the linear program. For example, if we set $(x_s, x_p) = (200, 700)$, then we have $200 + 700 \leq 1000$, $200 + 2 \times 700 \leq 1600$, $200 \geq 0$ and $700 \geq 0$. Thus, the solution $(200, 700)$ is a feasible solution to the linear program. This solution provides an objective value of $3 \times 200 + 4 \times 700 = 3400$. On the other hand, a pair of values for the decision variables (x_s, x_p) that maximizes the objective function, while satisfying all of the constraints is called an optimal solution. There is no feasible solution to the linear program that provides an objective value exceeding the objective value provided by the optimal solution. In certain problems, there can be multiple optimal solutions. We will come back to the possibility of multiple optimal solutions later on.

In a few lectures, we discuss algorithms to find an optimal solution to the linear program above. Before we go into these algorithms, we demonstrate how to use Microsoft Excel's solver to obtain an optimal solution. We set up a spreadsheet where two cells include the values of our decision variables. In the figure below, we use cells `A1` and `B1` to include the values of our decision variables. For the time being, we put dummy values of 1 and 1 into these cells. Next, we set up a formula that computes the objective function as a

function of the decision variables. We use the cell A2 for this purpose, where we include the formula = 3 * A1 + 4 * B1. Similarly, we set up formulas that compute the left side of the constraints as a function of the decision variables. We use the cells A3 and A4 to set up the formulas for the left sides of the first two constraints. In these cells, we include the formulas = A1 + B1 and = A1 + 2 * B1. We do not need to set up formulas to deal with the last two non-negativity constraints in the linear program above, since Microsoft Excel's solver has options to automatically enforce the non-negativity constraints. After setting up the formulas, the spreadsheet should look like the one in the figure below.



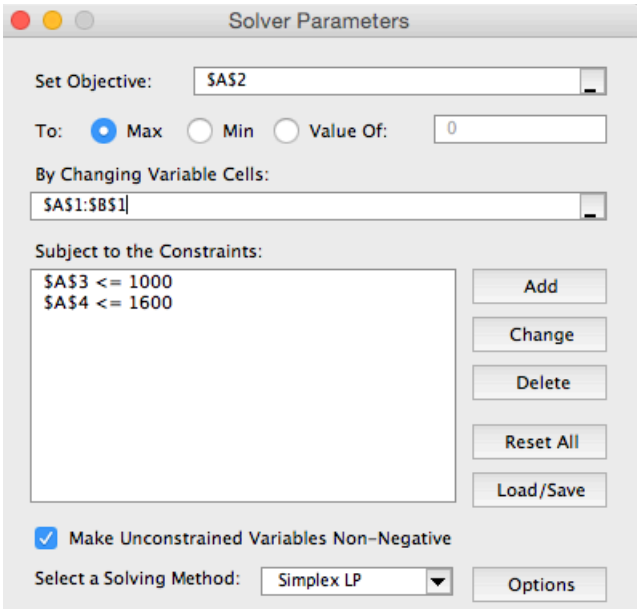
	A	B
1	1	1
2	=3*A1+4*B1	
3	=A1+B1	
4	=A1+2*B1	

Once we set up the formulas, we choose Solver under Tools menu. This action brings up a window titled Solver Parameters. In the box labeled Set Objective, we put the reference for the cell that includes the formula for the objective function, which is A2. In the box labeled To, we choose Max since we want to maximize the value of the objective function. In the box labeled By Changing Variable Cells, we put =$A$1:$B$1, which is the range of cells that includes our decision variables. Next, we click on Add to specify the constraints for our problem. This action brings up a window titled Add Constraints.

In the box labeled Cell Reference, we put A3, which includes the formula for the left side of the first constraint. In the middle box, we keep <=. In the box labeled Constraint, we put 1000, which is the right side of the first constraint. We click on Add, which adds the first constraint into the linear program. In the same way, we include the second constraint into the linear program. In particular, in the box labeled Cell Reference, we put A4, which includes the formula for the left side of the second constraint. In the middle box, we keep <=. In the box labeled Constraint, we put 1600, which is the right side of the second constraint. We click on OK to note that we added all of the constraints that we want to add. This action brings us back to the window titled Solver Parameters.

In this window, we make sure that Make Unconstrained Variables Non-Negative is checked so that the decision variables are constrained to be non-negative. In the drop down menu titled Select a Solving Method, we choose Simplex LP, which is the algorithm that is appropriate for solving linear programs. After constructing the linear program as described above, the window titled Solver Parameters should look like the one in the figure below. We click on Solve in the window titled Solver Parameters. Microsoft Excel's solver adjusts the values in the cells A1 and B1, which include the values of our decision variables. A dialog box appears to inform us that the optimal solution to the problem has been reached. The values in the cells A1 and B1 correspond to the optimal values of our decision variables. For

this problem, the optimal solution is given by $(x_s^*, x_p^*) = (400, 600)$. The objective value provided by this solution is $3 \times 400 + 4 \times 600 = 3600$.



2 Displaying Ads to Website Visitors

Assume that we have three advertisers, A, B and C, running ads on a website that we operate. Advertisers A, B and C would like their ads be seen by 2000, 3000 and 1000 viewers per day, respectively. There are 3 visitor types, 1, 2 and 3, that visit our website. These visitor types correspond to visitors in the age ranges $[20, 30)$, $[30, 40)$ and $[40, 50)$. The daily numbers of visitors of type 1, 2 and 3 that visit our website are 1500, 2000 and 2500, respectively. Each visitor sees at most one ad. If a visitor of a certain type sees an ad from a certain advertiser, then we generate a revenue that depends on the type of the visitor and the advertiser. These revenues are given in the following table. For example, we obtain a revenue of 2.5 when we show a visitor of type 1 an ad from advertiser C.

	A	B	C
1	1.5	3.5	2.5
2	2	1	3
3	1.5	4	2

We are interested in figuring out how many ads from each advertiser to show to how many viewers of each type to maximize the revenue per day. To formulate the problem as a linear program, we use the following decision variables.

We define seven other decision variables, $x_{1C}, x_{2A}, x_{2B}, x_{2C}, x_{3A}, x_{3B}$ and x_{3C} with similar interpretations. As a function of the decision variables, the revenue obtained per day is $1.5\,x_{1A} + 3.5\,x_{1B} + 2.5\,x_{1C} + 2\,x_{2A} + x_{2B} + 3\,x_{2C} + 1.5\,x_{3A} + 4\,x_{3B} + 2\,x_{3C}$, which is the objective function that we want to maximize. Constraints for this problem require a little bit more thought. We have 1500 viewers of type 1. The total number visitors of type 1 that are shown an ad from advertisers A, B and C cannot exceed the daily number of visitors of type 1. We can express this constraint as $x_{1A} + x_{1B} + x_{1C} \leq 1500$. By following the same reasoning, we can write constraints for visitors of type 2 and 3, which can be expressed as $x_{2A} + x_{2B} + x_{2C} \leq 2000$ and $x_{3A} + x_{3B} + x_{3C} \leq 2500$. One the other hand, advertiser A would like its ad be seen by 2000 viewers. Thus, the total number of visitors of types 1, 2 and 3, that are shown an ad from advertiser A should be 2000. We can express this constraint as $x_{1A} + x_{2A} + x_{3A} = 2000$. By following the same reasoning, we can write constraints for advertisers B and C, which can be expressed as $x_{1B} + x_{2B} + x_{3B} = 3000$ and $x_{1C} + x_{2C} + x_{3C} = 1000$. Naturally, we need constraints to ensure that all of our decision variables are non-negative. Putting it all together, the problem we are interested in can be formulated as the linear program

Considering practical applications, the linear program above is actually not that large. Many linear programs that appear in practical applications include thousands of decision variables and thousands of constraints. Explicitly listing all of the decision variables and the constraints in such linear programs can easily become tedious. To overcome this difficulty, we often express a linear program in compact form. We use the example above to

demonstrate how we can express a linear program in compact form. We represent the known data of the problem as follows.

R_{ij} = Revenue obtained by showing a visitor of type i an ad from advertiser j, $i = 1, 2, 3$, $j = A, B, C$.

V_i = Daily number of visitors of type i, $i = 1, 2, 3$.

D_j = Daily number of viewers desired by advertiser j, $j = A, B, C$.

For example, we have $R_{1A} = 1.5$, $R_{3B} = 4$, $V_1 = 1500$, $V_2 = 2000$, $V_3 = 2500$, $D_A = 2000$, $D_B = 3000$, $D_C = 1000$. Note that $\{R_{ij} : i = 1, 2, 3,\ j = A, B, C\}$, $\{V_i : i = 1, 2, 3\}$, $\{D_j : j = A, B, C\}$ are known data of the problem. Furthermore, we express the decision variables as follows.

x_{ij} = Number of visitors of type i that are shown an ad from advertiser j, $i = 1, 2, 3$, $j = A, B, C$.

As a function of our decision variables, we can write the daily revenue as $\sum_{i=1}^{3} \sum_{j=A}^{C} R_{ij}\, x_{ij}$. The total number of visitors of type i that are shown an ad from advertisers A, B and C is $\sum_{j=A}^{C} x_{ij}$. Since the total number of visitors of type i that are shown an ad from advertisers A, B and C cannot exceed the daily number of visitors of type i, we have the constraints $\sum_{j=A}^{C} x_{ij} \leq V_i$ for all $i = 1, 2, 3$. The total number of visitors of types 1, 2 and 3 that are shown an ad from advertiser j is $\sum_{i=1}^{3} x_{ij}$. Since the total number of visitors of types 1, 2 and 3 that are shown an ad from advertiser j should be equal to the daily number of viewers desired advertiser j, we have the constraints $\sum_{i=1}^{3} x_{ij} = D_j$ for all $j = A, B, C$. Putting it all together, the problem we are interested in can be formulated as the linear program

Geometry of Linear Programming

In this chapter, we build some intuition into how we can solve a linear program by understanding the geometry behind a linear programming problem.

1 Plotting the Set of Feasible Solutions

We use the following example to understand how to plot the set of feasible solutions of a linear program. Assume that we provide computing services to two types of customers, CPU-intensive and memory-intensive. We have a total of 1000 CPU's and 1200 GB of memory at our disposal. Each CPU-intensive customer uses 2 CPU's and 1 GB of memory for a whole day and pays \$3 per day. Each memory-intensive customer uses 1 CPU and 2 GB of memory for a whole day and pays \$4 per day. Due to energy limits, we cannot serve more than 400 CPU-intensive customers on a given day. We want to decide how many customers of each type to serve daily to maximize the revenue. To formulate the problem as a linear program, we use two decision variables defined as follows.

x_1 = Number of CPU-intensive customers that we serve per day.

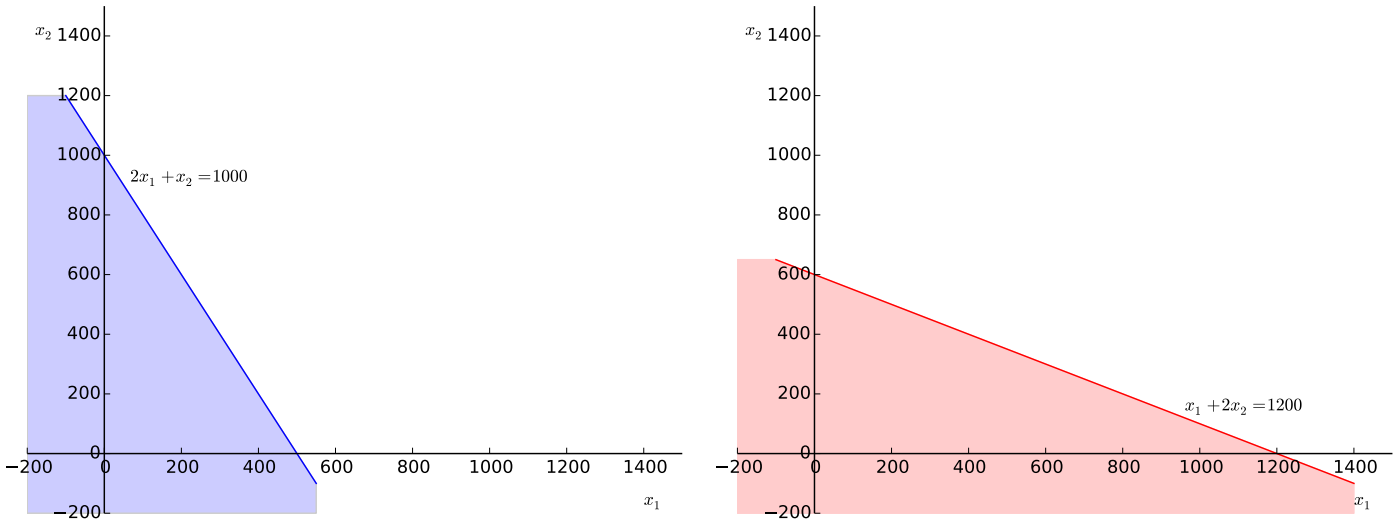
x_2 = Number of memory-intensive customers that we serve per day.

We can find the numbers of the two types customers to serve to maximize the revenue per day by solving the linear program

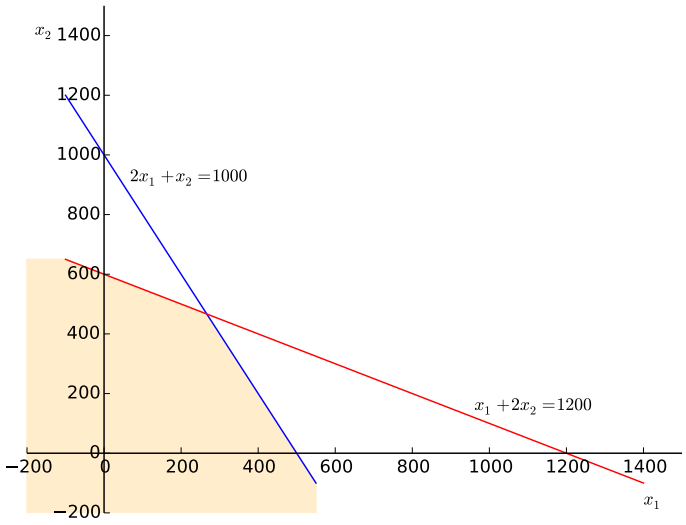
The objective function above accounts for the revenue that we obtain per day. The first constraint ensures that the total number of CPU's used by the two customer types on each day does not exceed the number of available CPU's. The second constraint ensures that the total amount of memory used by the two customer types on each day does not exceed the available memory. The third constraint ensures that we do not serve more than 400 CPU-intensive customers per day.

The set of (x_1, x_2) pairs that satisfy all of the constraints is called the set of feasible solutions to the linear program. To understand the set of feasible solutions to the linear program above, we plot the set of (x_1, x_2) pairs that satisfy each constraint. Consider the constraint $2\,x_1 + x_2 \leq 1000$. Note that $2\,x_1 + x_2 = 1000$ describes a line in the two-dimensional

plane. We plot this line in the left side of the figure below. The point $(x_1, x_2) = (0, 0)$ is to the lower left side of this line and it satisfies $2 \times 0 + 0 \leq 1000$. Therefore, all points to the lower left side of the line satisfy the constraint $2\,x_1 + x_2 \leq 1000$. We shade these points in light blue. Similarly, consider the constraint $x_1 + 2\,x_2 \leq 1200$. As before, $x_1 + 2\,x_2 = 1200$ describes a line in the two-dimensional plane. We plot this line in the right side of the figure below. The point $(x_1, x_2) = (0, 0)$ is to the lower left side of this line and it satisfies $0 + 2 \times 0 \leq 1200$. So, all points to the lower left side of the line satisfy the constraint $x_1 + 2\,x_2 \leq 1200$. We shade these points in light red.



In the figure below, we take the intersection of the light blue and light red regions in the previous figure, which means that the set of points that satisfy both of the constraints $2\,x_1 + x_2 \leq 1000$ and $x_1 + 2\,x_2 \leq 1200$ is given by the light orange region below.



Carrying out the same argument for the constraints $x_1 \leq 400$, $x_1 \geq 0$ and $x_2 \geq 0$, it follows that the set of points that satisfy all of the constraints in the linear program is given by the

region shaded in light green below.

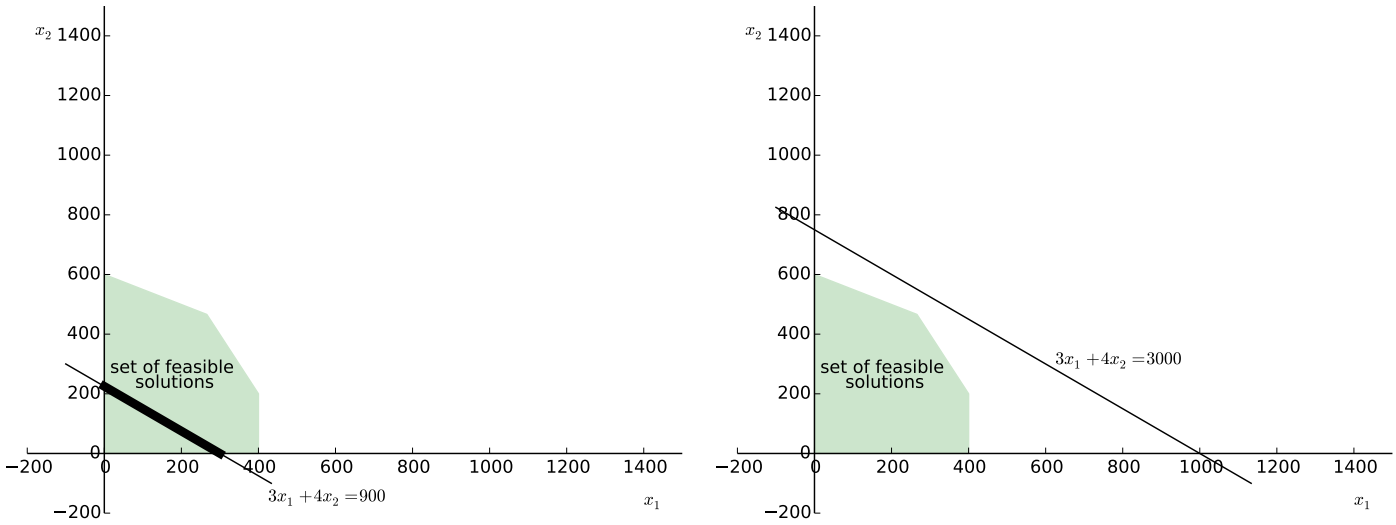
Any (x_1, x_2) pair in the light green region above is a feasible solution to our linear program. We want to find the feasible solution that maximizes the objective function.

2 Checking for a Target Objective Function Value

Before finding an optimal solution to the linear program, we consider the question of whether there exists a feasible solution that provides a certain target revenue. For example, consider the question of whether there exists a feasible solution that provides a revenue of 900. As a function of our decision variables, the revenue is given by $3\,x_1 + 4\,x_2$. So, we are interested in whether there exists (x_1, x_2) in the set of feasible solutions such that $3\,x_1 + 4\,x_2 = 900$. Note that $3\,x_1 + 4\,x_2 = 900$ describes a line in the two-dimensional plane. We plot this line in figure on the left side below in thin black. Thus, to check whether there exists (x_1, x_2) in the set of feasible solutions such that $3\,x_1 + 4\,x_2 = 900$, we need to check whether there are any points (x_1, x_2) that lie both in the set of feasible solutions and on the line $3\,x_1 + 4\,x_2 = 900$. In the figure on the left side below, the set of feasible solutions is given by light green region. Thus, there are indeed points that lie both in the set of feasible solutions and on the line $3\,x_1 + 4\,x_2 = 900$. These points are colored in thick black. In other words, since the intersection between the set of feasible solutions and the line $3\,x_1 + 4\,x_2 = 900$ is nonempty,

we can achieve a revenue of 900 by using a feasible solution to the linear program. So, the optimal revenue in the linear program must be at least 900.

Similarly, consider the question of whether there exists a feasible solution that provides a revenue of 3000. In other words, we are interested in whether there exists (x_1, x_2) in the set of feasible solutions such that $3\,x_1 + 4\,x_2 = 3000$. In the figure on the right side below, we plot the line $3\,x_1 + 4\,x_2 = 3000$ in black. We observe that there are no points that lie both in the set of feasible solutions and on the line $3\,x_1 + 4\,x_2 = 3000$. Therefore, there is no feasible solution that provides a revenue of 3000, since the intersection between the set of feasible solutions and the line $3\,x_1 + 4\,x_2 = 3000$ is empty.



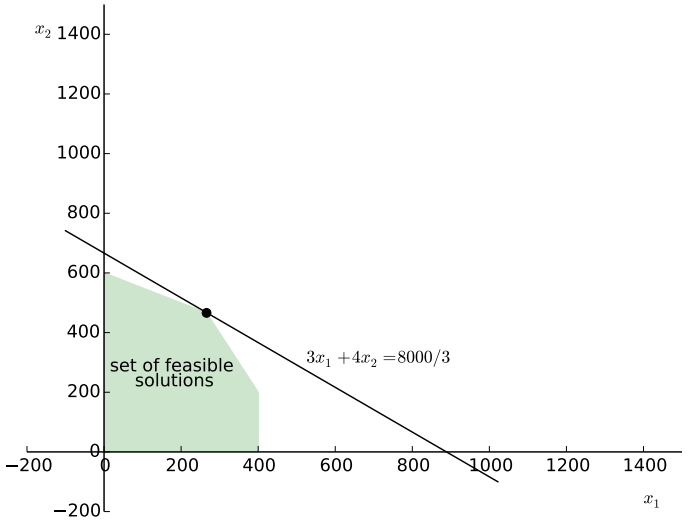
Observe that the lines $3\,x_1 + 4\,x_2 = 900$ and $3\,x_1 + 4\,x_2 = 3000$ are all parallel to each other. Therefore, to check whether there exists a feasible solution that provides a revenue of K, we can shift the line $3\,x_1 + 4\,x_2 = 900$ parallel to itself until we obtain the line $3\,x_1 + 4\,x_2 = K$. If the line $3\,x_1 + 4\,x_2 = K$ is still in contact with the set of feasible solutions, then there exists a feasible solution that provides a revenue of K.

3 Finding an Optimal Solution

From the discussion in the previous section, there exists a feasible solution that provides a revenue of 900, which was verified by showing that the intersection between the set of feasible solutions and the line $3\,x_1 + 4\,x_2 = 900$ is nonempty. So, the optimal revenue should be at least 900. Similarly, there does not exists a feasible solution that provides a revenue of 3000, which was verified by showing that the intersection between the set of feasible solutions and the line $3\,x_1 + 4\,x_2 = 3000$ is empty.

To find the optimal revenue, we need to find the largest value of K such that the intersection between the set of feasible solutions and the line $3\,x_1 + 4\,x_2 = K$ is nonempty. The lines $3\,x_1 + 4\,x_2 = K$ for different values of K are parallel to each other and these lines move

to the upper right side of the figure below as K increases. Thus, to find the optimal revenue, we need to move the line $3\,x_1 + 4\,x_2 = K$ to the upper right side as much as possible while making sure that the intersection between the set of feasible solutions and the line $3\,x_1 + 4\,x_2 = K$ is nonempty. This approach yields the line plotted in black in the figure below, which barely touches the set of feasible solutions at the black dot.

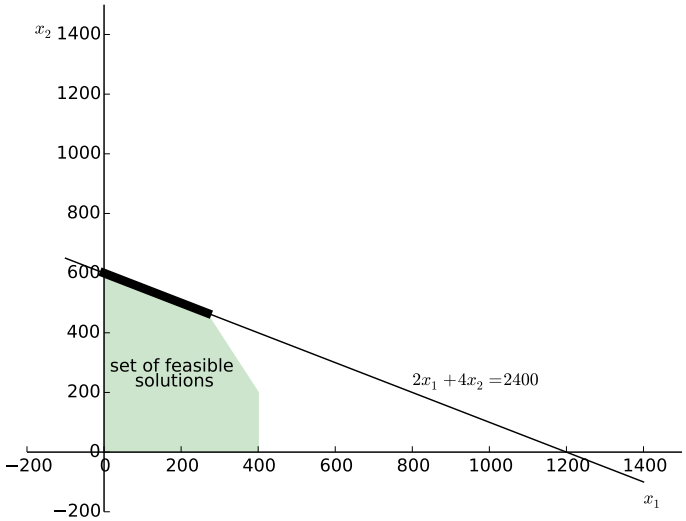


The coordinates of the black dot in the figure above gives the optimal solution to the problem. To compute these coordinates, we observe that the black dot lies on the lines that represent the first two constraints in the linear program and these lines are given by the equations $2\,x_1 + x_2 = 1000$ and $x_1 + 2\,x_2 = 1200$. Solving these two equations simultaneously, we obtain $x_1 = 800/3$ and $x_2 = 1400/3$. Therefore, the optimal solution to the linear program is given by $(x_1^*, x_2^*) = (800/3, 1400/3)$. The revenue from this solution is $3\,x_1^* + 4\,x_2^* = 3 \times \frac{800}{3} + 4 \times \frac{1400}{3} = \frac{8000}{3}$, which is the optimal objective value.

A key observations from the discussion above is that the optimal solution to a linear program is achieved at one of the corner points of the set of feasible solutions. This observation is critical for the following reason. There are infinitely many possible feasible solutions to the linear program. So, we cannot check the objective value provided by each possible feasible solution. However, there are only finitely many possible corner points of the set of feasible solutions. If we know that the optimal solution to a linear program occurs at one of the corner points, then we can check the objective value achieved at the corner points and pick the corner point that provides the largest objective value. Using this observation, we will develop an algorithm to efficiently solve linear programs when there are more than two decision variables and we cannot even plot the set of feasible solutions.

It is also useful to observe that in the optimal solution $(x_1^*, x_2^*) = (800/3, 1400/3)$, we have $x_1^* < 400$. Thus, the third constraint in the linear program does not play a role in determining the optimal solution, which implies that the optimal solution would not change even if we dropped this constraint from the linear program.

Assume for the moment that the objective function of the linear program were $2\,x_1 + 4\,x_2$, instead of $3\,x_1 + 4\,x_2$. We leave the constraints unchanged. In the figure below, we plot the line $2\,x_1 + 4\,x_2 = 2400$ and the set of feasible solutions. Notice that all points that are colored in thick black lie both in the set of feasible solutions and on the line $2\,x_1 + 4\,x_2 = 2400$. Furthermore, for any value of $K > 2400$, the intersection between the set of feasible solutions and the line $2\,x_1 + 4\,x_2 = K$ is empty. Therefore, the largest revenue that we can obtain is 2400 and any one of the points colored in thick black provides this revenue. In other words, if the objective function were $2\,x_1 + 4\,x_2$, then the linear program would have multiple optimal solutions and all of the points colored in thick black in the figure below would be an optimal solution.



Linear Algebra Concepts

In this chapter, we review some linear algebra concepts that will become useful when we develop an algorithm to solve linear programs.

1 Matrices and Vectors

An $m \times n$ matrix A is characterized by the entries $\{a_{ij} : i = 1, \ldots, m,\ j = 1, \ldots, n\}$, where a_{ij} is the entry in row i and column j. Thus, a matrix $A \in \Re^{m \times n}$ is represented as

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \ldots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \ldots & a_{2n} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \ldots & a_{mn} \end{bmatrix}.$$

We denote the transpose of matrix A by A^t. If $A = \{a_{ij} : i = 1, \ldots, m,\ j = 1, \ldots, n\} \in \Re^{m \times n}$, then $A^t = \{a_{ji} : j = 1, \ldots, n,\ i = 1, \ldots, m\} \in \Re^{n \times m}$. For example, if we have

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 7 \\ 1 & 4 & 2 \\ 4 & 5 & 1 \end{bmatrix}, \text{ then } A^t = \begin{bmatrix} 1 & 3 & 1 & 4 \\ 2 & 1 & 4 & 5 \\ 3 & 7 & 2 & 1 \end{bmatrix}.$$

A matrix $I = \{I_{ij} : i = 1, \ldots, n,\ j = 1, \ldots, n\} \in \Re^{n \times n}$ is called the $n \times n$ identity matrix if its diagonal entries are 1 and off-diagonal entries are 0. The 4×4 identity matrix is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

We refer to an $n \times 1$ matrix as a vector in $\Re^n$. A vector $x \in \Re^n$ is characterized by the entries $\{x_j : j = 1, \ldots, n\}$. Thus, a vector $x \in \Re^n$ is represented as

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

Note that we follow the convention that a vector is always a column vector, which is essentially a matrix with one column and multiple rows.

2 Matrix Addition and Multiplication

If we have two matrices that are of the same dimension, then we can add them. To demonstrate matrix addition, we have

$$\begin{bmatrix} 1 & 2 & -1 \\ 3 & 1 & 7 \\ 1 & 4 & 2 \\ 4 & 5 & 1 \end{bmatrix} + \begin{bmatrix} 2 & 4 & 9 \\ 1 & 3 & 8 \\ 7 & 7 & 1 \\ 2 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 6 & 8 \\ 4 & 4 & 15 \\ 8 & 11 & 3 \\ 6 & 4 & 1 \end{bmatrix}.$$

So, if $A = \{a_{ij} : i = 1, \ldots, m,\ j = 1, \ldots, n\}$ and $B = \{b_{ij} : i = 1, \ldots, m,\ j = 1, \ldots, n\}$, then $A + B = \{a_{ij} + b_{ij} : i = 1, \ldots, m,\ j = 1, \ldots, n\}$. Similar to addition, we can subtract two matrices that are of the same dimension. The only difference is that we subtract the corresponding entries rather than adding them.

We can multiply an $m \times r$ matrix with an $r \times n$ matrix to obtain an $m \times n$ matrix. If $A = \{a_{ik} : i = 1, \ldots, m,\ k = 1, \ldots, r\} \in \Re^{m \times r}$ and $B = \{b_{kj} : k = 1, \ldots, r,\ j = 1, \ldots, n\} \in \Re^{r \times n}$, then $AB = \{\sum_{k=1}^{r} a_{ik}\, b_{kj} : i = 1, \ldots, m,\ j = 1, \ldots, n\} \in \Re^{m \times n}$. To demonstrate matrix multiplication, we have

$$\begin{bmatrix} 1 & 3 & 4 & 2 \\ 2 & 4 & 1 & 3 \\ 5 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 2 & 3 \\ 1 & 1 \\ 1 & -1 \\ 4 & 2 \end{bmatrix} = \begin{bmatrix} 17 & 6 \\ 21 & 15 \\ 25 & 20 \end{bmatrix}.$$

To verify the computation above, note that the entry in row 1 and column 1 of the product matrix is $\sum_{k=1}^{4} a_{1k}\, b_{k1} = 1 \times 2 + 3 \times 1 + 4 \times 1 + 2 \times 4 = 17$. Similarly, the entry in row 3 column 2 of the product matrix is $\sum_{k=1}^{4} a_{3k}\, b_{k2} = 5 \times 3 + 1 \times 1 - 2 \times 1 + 3 \times 2 = 20$. The other entries can be verified in a similar fashion.

We can write a sum of the form $a_1\, x_1 + a_2\, x_2 + a_3\, x_3 + a_4 x_4$ as the product of two vectors. In particular, considering the vectors

$$a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} \quad \text{and} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix},$$

we have $a_1\, x_1 + a_2\, x_2 + a_3\, x_3 + a_4 x_4 = a^t x$. We can use matrix multiplication to represent a system of linear equations. For example, consider the system of equations

$$\begin{array}{rcrcrcrcl} 5\,x_1 & + & 6\,x_2 & + & 3\,x_3 & + & x_4 & = & 7 \\ 6\,x_1 & + & 2\,x_2 & & & + & 4\,x_4 & = & 8 \\ 9\,x_1 & & & + & 6\,x_3 & + & 2\,x_4 & = & 1. \end{array}$$

Using a matrix to represent the coefficients on the left side above, we can write this system of equations equivalently as

$$\begin{bmatrix} 5 & 6 & 3 & 1 \\ 6 & 2 & 0 & 4 \\ 9 & 0 & 6 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 7 \\ 8 \\ 1 \end{bmatrix}.$$

Matrix multiplication is not a commutative operation. So, we do not have $AB = BA$. If I is the identity matrix of appropriate dimensions, then $AI = A$ and $IA = A$. Also, we have $(AB)^t = B^t A^t$.

3 Matrix Inversion

For an $n \times n$ matrix A, we denote its inverse by A^{-1}. The inverse of an $n \times n$ matrix is also an $n \times n$ matrix. We have

$$AA^{-1} = A^{-1}A = I,$$

where I is the identity matrix. Computing the inverse of a matrix is related to row operations. Consider computing the inverse of the matrix

$$A = \begin{bmatrix} 1 & 2 & 2 \\ 2 & -1 & 1 \\ 1 & 0 & 2 \end{bmatrix}.$$

To compute the inverse of this matrix, we augment this matrix with the identity matrix on the right side to obtain the matrix

A row operation refers to multiplying one row of a matrix with a constant or adding a multiple of one row to another row. To compute the inverse of the 3×3 matrix A above, we carry out a sequence of row operations on the matrix $[A \,|\, I]$ to bring $[A \,|\, I]$ in the form of $[I \,|\, B]$. In this case, B is the inverse of A. Consider the matrix $[A \,|\, I]$ given by

Multiply the first row by -2 and add to the second row. Also, multiply the first row by -1 and add to the third row. Thus, we get

Multiply the second row by $-1/5$. Thus, we get

Multiply the second row by -2 and add to the first row. Also, multiply the second row by 2 and add to the third row. Thus, we get

Multiply the third row by $5/6$. Thus, we get

Multiply the third row by $-4/5$ and add to the first row. Also, multiply the third row by $-3/5$ and add to the second row. Thus, we get

Simplifying the fractions, we have

Note that the last matrix above is of the form $[I \mid B]$. Therefore, it follows that the inverse of the matrix

$$\begin{bmatrix} 1 & 2 & 2 \\ 2 & -1 & 1 \\ 1 & 0 & 2 \end{bmatrix} \quad \text{is} \quad \begin{bmatrix} \frac{1}{3} & \frac{2}{3} & -\frac{2}{3} \\ \frac{1}{2} & 0 & -\frac{1}{2} \\ -\frac{1}{6} & -\frac{1}{3} & \frac{5}{6} \end{bmatrix}.$$

To check that our computations are correct, we can multiply the two matrices above to see that we get the identity matrix. In particular, we have

$$\begin{bmatrix} 1 & 2 & 2 \\ 2 & -1 & 1 \\ 1 & 0 & 2 \end{bmatrix} \begin{bmatrix} \frac{1}{3} & \frac{2}{3} & -\frac{2}{3} \\ \frac{1}{2} & 0 & -\frac{1}{2} \\ -\frac{1}{6} & -\frac{1}{3} & \frac{5}{6} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Matrix inversion is useful to solve a system of linear equations. To demonstrate, consider the system of equations

$$\begin{aligned} x_1 + 2\,x_2 + 2\,x_3 &= 12 \\ 2x_1 - x_2 + x_3 &= 4 \\ x_1 + 2\,x_3 &= 18, \end{aligned}$$

which is equivalent to

$$\begin{bmatrix} 1 & 2 & 2 \\ 2 & -1 & 1 \\ 1 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 12 \\ 4 \\ 8 \end{bmatrix}.$$

Using $A \in \Re^{3\times 3}$ to denote the matrix on the left side, $x \in \Re^{3\times 1}$ to denote the vector on the left side and $b \in \Re^{3\times 1}$ to denote the vector on the right side, the equation above is of the form $A\,x = b$. Multiplying both side of the this equality by A^{-1}, we get

$$A^{-1}\,A\,x = A^{-1}\,b \implies I\,x = A^{-1}\,b \implies x = A^{-1}\,b.$$

Thus, the solution to the system of equations above is given by $x = A^{-1}\,b$.

We emphasize that not every matrix has an inverse.

4 Systems of Equations and Row Operations

Consider the set of points that satisfy the system of equations $2\,x_1 + x_2 = 4$ and $-x_1 + x_2 = 1$. Each of these equations characterizes a line in the two-dimensional plane. Plotting these lines in the figure below, we observe that the set of points that satisfy the system of equations $2\,x_1 + x_2 = 4$ and $-x_1 + x_2 = 1$ is a single point given by $(x_1, x_2) = (1, 2)$.

We apply an arbitrary sequence of row operations on the system of equations

$$\begin{array}{rcrcl} 2\,x_1 & + & x_2 & = & 4 \\ -x_1 & + & x_2 & = & 1. \end{array}$$

For example, add the first row to the second row to get

$$\begin{array}{rcrcl} 2\,x_1 & + & x_2 & = & 4 \\ x_1 & + & 2\,x_2 & = & 5. \end{array}$$

Multiply the second row by $-4/5$ and add to the first row to get

$$\begin{array}{rcrcl} \frac{6}{5}\,x_1 & - & \frac{3}{5}x_2 & = & 0 \\ x_1 & + & 2\,x_2 & = & 5. \end{array}$$

Noting the last system of equations above, consider the set of points that satisfy the system of equations $\frac{6}{5}\,x_1 - \frac{3}{5}\,x_2 = 0$ and $x_1 + 2\,x_2 = 5$. Plotting the lines that are characterized by each of these equations in the figure below, we observe that the set of points that satisfy the system of equations $\frac{6}{5}\,x_1 - \frac{3}{5}\,x_2 = 0$ and $x_1 + 2\,x_2 = 5$ is the single point given by $(x_1, x_2) = (1, 2)$. Observe that this is the same point that satisfy the system of equations that we started with. Therefore, the critical observation from this discussion is that the set of points that satisfy a system of equations does not change when we apply any sequence of row operations to a system of equations.

We keep on applying row operations on the last system of equations, which is given by

$$\begin{array}{rcrcl} \frac{6}{5}\,x_1 & - & \frac{3}{5}x_2 & = & 0 \\ x_1 & + & 2\,x_2 & = & 5. \end{array}$$

Multiply the first row by $-5/6$ and add to the second row to get

$$\begin{array}{rcrcl} \frac{6}{5}\,x_1 & - & \frac{3}{5}\,x_2 & = & 0 \\ & + & \frac{5}{2}\,x_2 & = & 5. \end{array}$$

Multiply the second row by $6/25$ and add to the first row to get

$$\begin{array}{rcrcl} \frac{6}{5}\,x_1 & & & = & \frac{6}{5} \\ & & \frac{5}{2}\,x_2 & = & 5. \end{array}$$

Multiply the first row by $5/6$ and the second row by $2/5$ to get

$$\begin{array}{rcrcl} x_1 & & & = & 1 \\ & & x_2 & = & 2. \end{array}$$

The last system of equations is obtained by applying a sequence of row operations on the original system of equations. We know that the set of points that satisfy a system of equations does not change when we apply any row operations to the system of equations. Thus, the set of points that satisfy the original system of equations

$$\begin{array}{rcrcl} 2\,x_1 & + & x_2 & = & 4 \\ -x_1 & + & x_2 & = & 1 \end{array}$$

is the same as the set of points that satisfy the last system of equations

$$\begin{aligned} x_1 &= 1 \\ x_2 &= 2. \end{aligned}$$

We can immediately see that the set of points that satisfy the last system of equations is the single point $(x_1, x_2) = (1, 2)$. Therefore, the set of points that satisfy the original system of equations is also the single point $(x_1, x_2) = (1, 2)$. This discussion shows why row operations are useful to solve a system of equations.

Simplex Method for Solving Linear Programs

Understanding the geometry of linear programs allowed us to solve small linear programs by using graphical methods. However, this approach becomes ineffective when the number of decision variables exceeds two or three. In this chapter, we develop the simplex method for solving large linear programs.

1 Key Idea of the Simplex Method

Consider the linear program

We want to solve this linear program without using graphical methods. The first thing we do is to introduce the decision variable w_1 to represent how much the right side of the first constraint above exceeds the left side of the constraint. That is, we have $w_1 = 1000 - 2\,x_1 - x_2$. If (x_1, x_2) is a feasible solution to the linear program above, then we must have $w_1 \geq 0$ and $w_1 = 1000 - 2\,x_1 - x_2$. We refer to w_1 as the slack variable associated with the first constraint. Similarly, we associate the slack variable w_2 with the second constraint so that $w_2 = 1200 - x_1 - 2\,x_2$. Thus, if (x_1, x_2) is a feasible solution to the linear program above, then we must have $w_2 \geq 0$ and $w_2 = 1200 - x_1 - 2\,x_2$. Finally, we associate the slack variable w_3 with the third constraint so that $w_3 = 400 - x_1$. So, if (x_1, x_2) is a feasible solution to the linear program above, then we must have $w_3 \geq 0$ and $w_3 = 400 - x_1$. In this case, we can write the linear program above equivalently as

Since the two linear programs above are equivalent to each other, we focus on solving the second linear program. The advantage of the second linear program is that its constraints are of equality form. The simplex method expresses the system of equations associated with the second linear program above as

$$\begin{array}{rcrcrcrcrcr}
3\,x_1 & + & 4\,x_2 & & & & & & & = & z \\
\hline
2\,x_1 & + & x_2 & + & w_1 & & & & & = & 1000 \\
x_1 & + & 2\,x_2 & & & + & w_2 & & & = & 1200 \\
x_1 & & & & & & & + & w_3 & = & 400,
\end{array}$$

where the first row corresponds to the objective function and the other three rows correspond to the three constraints. The system of equations captures all the information that we have on the linear program. We do not explicitly express the non-negativity constraints on the decision variables, but we always keep in mind that all of the decision variables are constrained to be non-negative.

We make two observations for the system of equations above. First, if we keep on applying row operations to the system of equations, then the system of equations that we obtain through the row operations remain equivalent to the original system of equations. Second, the decision variable w_1 appears only in the first constraint row with a coefficient of 1 and nowhere else. Similarly, w_2 and w_3 respectively appear only in the second and third constraint rows with a coefficient of 1 and nowhere else. Thus, it is simple to spot a solution $(x_1, x_2, w_1, w_2, w_3)$ and z to the system of equations above. We can set

$$w_1 = 1000,\ w_2 = 1200,\ w_3 = 400,\ x_1 = 0,\ x_2 = 0,\ z = 0.$$

Note that the solution above is feasible to the linear program. Also note that the value of the decision variable z corresponds to the value of the objective function provided by the solution above. Now, we iteratively apply row operations to the system of equations above to obtain other solutions that are feasible to the linear program and provide larger objective function values. As we apply the row operations, we make sure that there is always a set of three variables such that each one of these variables appear in only one constraint row with a coefficient of 1. Furthermore, these variables do not appear in the objective function row and each of these three variables appear in a different constraint row. For example, in the system of equations above, each one of the three decision variables w_1, w_2 and w_3 appear in different constraint rows with a coefficient of 1 and they do not appear in the objective function row. We start with the system of equations

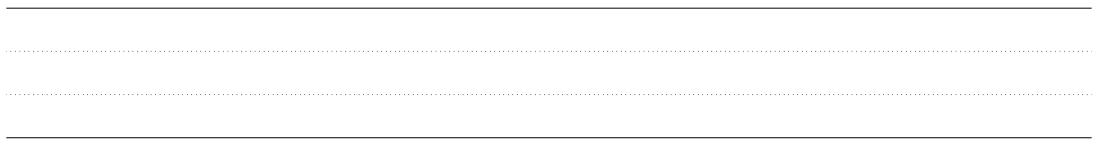
For this system of equations, we have the solution $w_1 = 1000,\ w_2 = 1200,\ w_3 = 400,\ x_1 = 0,\ x_2 = 0,\ z = 0$. From the objective function row, for each unit of increase in the decision variable x_1, the objective function increases by 3 units, whereas for each unit of increase in the decision variable x_2, the objective function increases by 4 units. Therefore, we will increase the value of the decision variable x_2.

The next question is how much we can increase the value of the decision variable x_2 while making sure that the solution on hand remains feasible and all of the decision variables remain non-negative. Considering the first constraint row above, w_1 is the decision variable that appears only in this row. Thus, if we increase x_2, then we can make up for the increase in x_2 by a decrease in w_1 to make sure that the first constraint remains satisfied. However, if we increase x_2 too much, then the decision variable w_1 may have to go negative. Note that we can increase x_2 up to 1000, while making sure that w_1 remains non-negative.

Similarly, considering the second constraint row, w_2 is the decision variable that appears only in this row. Thus, if we increase x_2, then we can make up for the increase in x_2 by a decrease in w_2 to make sure that the second constraint remains satisfied. Again, if we increase x_2 too much, then the decision variable w_2 may go negative. Note that we can increase x_2 up to 600, while making sure that w_2 remains non-negative.

Lastly, the decision variable x_2 does not appear in the third constraint row above. Therefore, we can increase x_2 as much as we want and the third constraint would remain satisfied. Considering the preceding discussion, since $\min\{1000, 600\} = 600$, we can increase x_2 at most up to 600 while making sure that all of the constraints remain satisfied and all of the decision variables remain non-negative.

If we increase x_2 up to 600, then the new value of the decision variable x_2 is determined by the second constraint row. Thus, we carry out row operations in the system of equations above to make sure that x_2 appears only in the second constraint row with a coefficient of 1. In particular, we multiply the second constraint row by -2 and add it to the objective function row. We multiply the second constraint row by $-1/2$ and add it to the first constraint row. Finally, we multiply the second constraint row by $1/2$. Through these row operations, we obtain the system of equations



We just carried row operations. So, the system of equations above is equivalent to the original one. Each one of the decision variables w_1, x_2 and w_3 appears only in each one of the three constraint rows above and nowhere else. Thus, it is simple to spot a solution

$(x_1, x_2, w_1, w_2, w_3)$ and z to the system of equations above. We can set

$$w_1 = 400,\ x_2 = 600,\ w_3 = 400,\ x_1 = 0,\ w_2 = 0,\ z = 2400.$$

The solution above is feasible to the linear program. It is actually not surprising that this solution is feasible, since this solution is obtained from the original system of equations by using row operations. Furthermore, the value of the decision variable z corresponds to the value of the objective function provided by the solution above. From the objective function row, for each unit of increase in the decision variable x_1, the objective function increases by 1 unit, whereas for each unit of increase in the decision variable w_2, the objective function decreases by 2 units. Therefore, we will increase the value of x_1.

Next, we ask how much we can increase the value of the decision variable x_1 while making sure that the other variables stay non-negative. Considering the first constraint row above, w_1 is the decision variable that appears only in this row. Thus, if we increase x_1, then we will make up for the increase in x_1 by a decrease in w_1. We can increase x_1 up to $400/\frac{3}{2} = 800/3$, while making sure that w_1 remains non-negative.

Considering the second constraint row above, x_2 is the decision variable that appears only in this row. Thus, if we increase x_1, then we will make up for the increase in x_1 by a decrease in x_2. We can increase x_1 up to $600/\frac{1}{2} = 1200$, while making sure that x_2 remains non-negative.

Considering the third constraint row above, w_3 is the decision variable that appears only in this row. Thus, if we increase x_1, then we will make up for the increase in x_1 by a decrease in w_3. We can increase x_1 up to 400, while making sure that w_3 remains non-negative. Since $\min\{800/3, 1200, 400\} = 800/3$, we can increase x_1 at most up to $800/3$ while making sure that all of the variables remain non-negative and all of the constraints remain satisfied.

When we increase x_1 up to $800/3$, the new value of the decision variable x_1 is determined by the first constraint. Therefore, we carry out row operations in the system of equations above to make sure that x_1 appears only in the first constraint row with a coefficient of 1. So, we multiply the first constraint row by $-2/3$ and add it to the objective row. We multiply the first constraint row by $-1/3$ and add it to the second constraint row. We multiply the first constraint row by $-2/3$ and add it to the third constraint row. Finally, we multiply the first constraint row by $2/3$. These row operations yield the system of equations

In the system of equations above, since each one of the decision variables $x_1,\ x_2$ and w_3 appears only in each of the three constraints above, the solution $(x_1, x_2, w_1, w_2, w_3)$ and z corresponding to the system of equations above is

$$x_1 = \frac{800}{3},\ x_2 = \frac{1400}{3},\ w_3 = \frac{400}{3},\ w_1 = 0,\ w_2 = 0,\ z = \frac{8000}{3}.$$

The solution above is a feasible solution to the linear program. Furthermore, from the objective function row of the last system of equations, we observe that increasing the value of one of the decision variables w_1 and w_2 decreases the objective function. So, we stop and conclude that the last solution above is an optimal solution to the linear program. In other words, the solution $(x_1, x_2, w_1, w_2, w_3) = (\frac{800}{3}, \frac{1400}{3}, 0, 0, \frac{400}{3})$ is an optimal solution providing the optimal objective value $\frac{8000}{3}$ for the linear program.

Throughout the iterations of the simplex method, we visited three solutions. The first solution is $(x_1, x_2, w_1, w_2, w_3) = (0, 0, 1000, 1200, 400)$ with an objective value of 0. The second solution is $(x_1, x_2, w_1, w_2, w_3) = (0, 600, 400, 0, 400)$ with an objective value of 2400. The third solution is $(x_1, x_2, w_1, w_2, w_3) = (\frac{800}{3}, \frac{1400}{3}, 0, 0, \frac{400}{3})$ with an objective value of $\frac{8000}{3}$. Therefore, at each iteration of the simplex method, we improve the objective value provided by the current solution. In the figure below, we show the set of feasible solutions to the linear program and the pairs (x_1, x_2) corresponding to each of the solutions visited by the simplex method. Note that the solutions visited by the simplex method correspond to the corner points of the set of feasible solutions.

2 Some Observations and Terminology

At each iteration, the simplex method visits a feasible solution to the linear program. We progressively increase the value of the objective function at each iteration. In particular, at the beginning of each iteration, we inspect the equation in the objective function row, pick the variable with the largest positive coefficient in this row and increase the value of this decision variable. Picking any decision variable with a positive coefficient in objective function row would be enough to increase the objective function value from one iteration to the next. Picking the variable with the largest positive coefficient is a heuristic for obtaining the largest increase in the objective function value at each iteration, but it does not necessarily guarantee that the simplex method will find the optimal solution in the quickest possible manner. Sometimes picking a variable with a positive, but not the largest positive, coefficient may allow finding the optimal solution more quickly.

Assume that we have m constraints and n decision variables in the original linear program with inequality constraints. Once we add the slack variables, we have a total of $n+m$ decision variables. At each iteration of the simplex method, there exists a set of m decision variables such that each one of these decision variables appears with a coefficient of 1 in only one constraint row and a coefficient of 0 in all other constraint rows and in the objective function row. We refer to these decision variables as basic variables. The remaining decision variables are referred to as non-basic variables. If we have m constraints and n decision variables in the original linear program, then we have m basic variables and n non-basic variables at each iteration. We emphasize that each basic variable appears with a coefficient of 1 in a different constraint row. To give an example, after applying the first set of row operations in the previous section, we obtained the system of equations

$$\begin{array}{rcrcrcrcrcl}
x_1 & & & & & - & 2\,w_2 & & & = & z - 2400 \\
\hline
\frac{3}{2}\,x_1 & & & + & w_1 & - & \frac{1}{2}\,w_2 & & & = & 400 \\
\frac{1}{2}\,x_1 & + & x_2 & & & + & \frac{1}{2}\,w_2 & & & = & 600 \\
x_1 & & & & & & & + & w_3 & = & 400.
\end{array}$$

In the system of equations above, the basic variables are w_1, x_2 and w_3. The non-basic variables are x_1 and w_2. The non-basic variables always take the value of zero. The values of the basic variables are given by the right side of the constraint rows. There is one constraint row that is associated with each basic variable. For example, the first constraint row above is associated with the basic variable w_1, whereas the second constraint row above is associated with the basic variable x_2.

As an alternative way to obtain the values of the basic variables at each iteration, we can go back to the equality constraints in the original linear program, set the values of the non-basic variables to zero and solve for the values of the m basic variables by using the m constraints. For example, in the system of equations above, the basic variables are w_1, x_2 and w_3, whereas the non-basic variables are x_1 and w_2. To obtain the values of the basic variables, we can go back to the equality constraints in the original linear program and set

$x_1 = 0$ and $w_2 = 0$ to obtain the system of equations $x_2 + w_1 = 1000$, $2\,x_2 = 1200$ and $w_3 = 400$. Solving this system of equations, we obtain $x_2 = 600$, $w_1 = 400$ and $w_3 = 400$, which is precisely the solution given by the right side of the constraint rows in the system of equations above.

At each iteration, one decision variable that was non-basic before becomes basic and one decision variable that was basic before becomes non-basic. For example, in the system of equations above, the basic variables are w_1, x_2 and w_3, whereas the non-basic variables are x_1 and w_2. If we apply row operations to make sure that x_1 appears only in the first constraint row with a coefficient of 1, then we obtain the system of equations

$$\begin{array}{rrrrrrrrrrr}
 & & & - & \frac{2}{3}\,w_1 & - & \frac{5}{3}\,w_2 & & & = & z - \frac{8000}{3} \\
\hline
x_1 & & & + & \frac{2}{3}\,w_1 & - & \frac{1}{3}\,w_2 & & & = & \frac{800}{3} \\
 & + & x_2 & - & \frac{1}{3}\,w_1 & + & \frac{2}{3}\,w_2 & & & = & \frac{1400}{3} \\
 & & & - & \frac{2}{3}\,w_1 & + & \frac{1}{3}\,w_2 & + & w_3 & = & \frac{400}{3}.
\end{array}$$

In the system of equations above, the basic variables are x_1, x_2 and w_3, whereas the non-basic variables are w_1 and w_2. Thus, through the row operations that we applied, the variable x_1 became basic and the variable w_1 became non-basic. At any iteration, the variable that becomes basic is called the entering variable. The variable that becomes non-basic is called the leaving variable. Solutions with m basic variables and n non-basic variables are called basic solutions. The solutions visited by the simplex method are basic solutions.

3 Simplex Method Applied on a Larger Example

Consider the linear program

$$\begin{array}{rl}
\max & 5\,x_1 + 3\,x_2 - x_3 \\
\text{st} & 4\,x_1 - x_2 + x_3 \leq 6 \\
 & 3\,x_1 + 2\,x_2 + x_3 \leq 9 \\
 & 4\,x_1 + x_2 - x_3 \leq 3 \\
 & x_1, x_2, x_3 \geq 0.
\end{array}$$

Using the slack variables, w_1, w_2 and w_3, we can write the linear program above as

Thus, we start with the system of equations

$$\begin{array}{rcrcrcrcrcrcl}
5\,x_1 & + & 3\,x_2 & - & x_3 & & & & & & & = & z \\
\hline
4\,x_1 & - & x_2 & + & x_3 & + & w_1 & & & & & = & 6 \\
3\,x_1 & + & 2\,x_2 & + & x_3 & & & + & w_2 & & & = & 9 \\
4\,x_1 & + & x_2 & - & x_3 & & & & & + & w_3 & = & 3,
\end{array}$$

In the system of equations above, the basic variables are $w_1,\ w_2$ and w_3, whereas the non-basic variables are $x_1,\ x_2$ and x_3. The values of the variables are given by

$$w_1 = 6,\ w_2 = 9,\ w_3 = 3,\ x_1 = 0,\ x_2 = 0,\ x_3 = 0,\ z = 0.$$

From the objective row, we observe that each unit of increase in x_1 increases the objective function by 5 units. Each unit of increase in x_2 increases the objective function by 3 units. Each unit of increase in x_3 decreases the objective function by 1 unit. Thus, we will increase the value of x_1.

Considering the first constraint row, w_1 is the decision variable that appears only in this row. Thus, if we increase x_1, then we will make up for the increase in x_1 by a decrease in w_1. We can increase x_1 up to $6/4$, while making sure that w_1 remains non-negative. Considering the second constraint row, w_2 is the decision variable that appears only in this row. Thus, if we increase x_1, then we will make up for the increase in x_1 by a decrease in w_2. We can increase x_1 up to $9/3 = 3$, while making sure that w_2 remains non-negative. Considering the third constraint row, w_3 is the decision variable that appears only in this row. Thus, if we increase x_1, then we will make up for the increase in x_1 by a decrease in w_3. We can increase x_1 up to $3/4$, while making sure that w_3 remains non-negative. By the preceding discussion, since $\min\{6/4, 3, 3/4\} = 3/4$, we can increase x_1 up to $3/4$ while making sure that all of the other decision variables remain non-negative.

If we increase x_1 up to $3/4$, then the new value of x_1 is determined by the third constraint row. Thus, we carry out row operations in the system of equations above to make sure that x_1 appears only in the third constraint row with a coefficient of 1. In other words, the entering variable is x_1 and the leaving variable is w_3. So, we multiply the third constraint row by $-5/4$ and add it to the objective function row. We multiply the third constraint row by -1 and add it to the first constraint row. We multiply the third constraint row by $-3/4$ and add it to the second constraint row. Finally, we multiply the third constraint row by $1/4$. In this case, we obtain the system of equations

The basic variables above are $w_1,\ w_2$ and x_1. The non-basic variables are $x_2,\ x_3$ and w_3. The values of the variables are given by

$$w_1 = 3,\ w_2 = \frac{27}{4},\ x_1 = \frac{3}{4},\ x_2 = 0,\ x_3 = 0,\ w_3 = 0,\ z = \frac{15}{4}.$$

We will now increase the value of x_2 since each unit of increase in x_2 increases the objective function by $7/4$ units.

Considering the first constraint row, w_1 is the decision variable that appears only in this row. We also observe that x_2 appears with a negative constraint coefficient in the first constraint row. Thus, if we increase x_2, then we can make up for the increase in x_2 by increasing w_1 to make sure that the first constraint remains satisfied. Thus, we can increase x_2 as much as we want without running into the danger of w_1 going negative, which implies that the first constraint row does not impose any restrictions on how much we can increase the value of x_2.

Considering the second constraint row, w_2 is the decision variable that appears only in this row. Thus, if we increase x_2, then we can make up for the increase in x_2 by a decrease in w_2. We can increase x_2 up to $\frac{27}{4}/\frac{5}{4} = 27/5$, while making sure that w_2 remains non-negative. Considering the third constraint row, x_1 is the decision variable that appears only in this row. Thus, if we increase x_2, then we can make up for the increase in x_2 by a decrease in x_1. We can increase x_2 up to $\frac{3}{4}/\frac{1}{4} = 3$, while making sure that x_1 remains non-negative. By the preceding discussion, since $\min\{27/5, 3\} = 3$, we can increase x_2 up to 3.

If we increase x_2 up to 3, then the new value of x_2 is determined by the third constraint row. Thus, we carry out row operations in the system of equations above to make sure that x_2 appears only in the third constraint row with a coefficient of 1. In other words, the entering variable is x_2 and the leaving variable is x_1. So, we multiply the third constraint row by -7 and add it to the objective function row. We multiply the third constraint row by 8 and add it to the first constraint row. We multiply the third constraint row by -5 and add it to the second constraint row. Finally, we multiply the third constraint row by 4. In this case, we obtain the system of equations

The basic variables above are $w_1,\ w_2$ and x_2. The non-basic variables are $x_1,\ x_3$ and w_3. The values of the variables are given by

$$w_1 = 9,\ w_2 = 3,\ x_2 = 3,\ x_1 = 0,\ x_3 = 0,\ w_3 = 0,\ z = 9.$$

We will increase the value of x_3 since each unit of increase in x_3 increases the objective function by 2 units.

Considering the first constraint row, x_3 does not appear in this row. Thus, we can increase x_3 as much as we want and the first constraint would remain satisfied. Considering the second constraint row, w_2 is the decision variable that appears only in this row. Thus, if we increase x_3, then we can make up for the increase in x_3 with a decrease in w_2. We can increase x_3 up to $3/3 = 1$ while making sure that w_2 remains non-negative.

Considering the third constraint row, x_2 is the decision variable that appears only in this row. We also observe that x_3 appears with a negative constraint coefficient in the third constraint row. Thus, if we increase x_3, then we can make up for the increase in x_3 by increasing x_2 to make sure that the third constraint remains satisfied. Thus, we can increase x_3 as much as we want without running into the danger of x_2 going negative, which implies that the third constraint row does not impose any restrictions on how much we can increase the value of x_3. By the preceding discussion, we can increase x_3 up to 1.

If we increase x_3 up to 1, then the new value of x_3 is determined by the second constraint row. Thus, we carry out row operations in the system of equations above to make sure that x_3 appears only in the second constraint row with a coefficient of 1. In other words, the entering variable is x_3 and the leaving variable is w_2. We multiply the second constraint row by $-2/3$ and add it to the objective function row. We multiply the second constraint row by $1/3$ and add it to the third constraint row. Finally, we multiply the second constraint row by $1/3$. So, we obtain the system of equations

The basic variables above are $w_1,\ x_3$ and x_2. The non-basic variables are $x_1,\ w_2$ and w_3. The values of the variables are given by

$$w_1 = 9,\ x_3 = 1,\ x_2 = 4,\ x_1 = 0,\ w_2 = 0,\ w_3 = 0,\ z = 11.$$

From the last system of equations, we observe that increasing the value of one of the decision variables x_1, w_2 and w_3 decreases the objective function value, since these variables have negative coefficients in the objective function row. So, we stop and conclude that the last solution above is an optimal solution to the linear program. In other words, the solution $(x_1, x_2, x_3, w_1, w_2, w_3) = (0, 4, 1, 9, 0, 0)$ is an optimal solution providing the optimal objective value 11 for the linear program.

4 Simplex Method in General Form

In this section, we describe the steps of the simplex method for a general linear program. Consider a linear program of the form

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} c_j\, x_j \\ \text{st} \quad & \sum_{j=1}^{n} a_{ij}\, x_j \le b_i \quad \forall\, i = 1, \ldots, m \\ & x_j \ge 0 \qquad\qquad \forall\, j = 1, \ldots, n. \end{aligned}$$

In the linear program above, there are n decision variables given by $x_1, \ldots, x_n$. The objective function coefficient of decision variable x_j is c_j. There m constraints. The right side coefficient of the i-th constraint is given by b_i. The decision variable x_j has the coefficient a_{ij} in the left side of the i-th constraint. Using the slack variables $w_1, \ldots, w_m$, we write the linear program above equivalently as

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} c_j\, x_j \\ \text{st} \quad & \sum_{j=1}^{n} a_{ij}\, x_j + w_i = b_i \quad \forall\, i = 1, \ldots, m \\ & x_j \ge 0, w_i \ge 0 \qquad\qquad \forall\, j = 1, \ldots, n,\ i = 1, \ldots, m. \end{aligned}$$

So, the simplex method starts with the system of equations

$$\begin{array}{rcrcccrcccrcr} c_1\, x_1 & + & c_2\, x_2 & + & \ldots & + & c_n\, x_n & & & & & = & z \\ \hline a_{11}\, x_1 & + & a_{12}\, x_2 & + & \ldots & + & a_{1n}\, x_n & + & w_1 & & & = & b_1 \\ a_{21}\, x_1 & + & a_{22}\, x_2 & + & \ldots & + & a_{2n}\, x_n & & & + \ w_2 & & = & b_2 \\ \vdots & & \vdots & & & & \vdots & & & & & = & \vdots \\ a_{m1}\, x_1 & + & a_{m2}\, x_2 & + & \ldots & + & a_{mn}\, x_n & & & & + \ w_m & = & b_m. \end{array}$$

To make our notation uniform, we label the variables $w_1, \ldots, w_m$ as $x_{n+1}, \ldots, x_{n+m}$, in which case the system of equations above looks like

$$\begin{array}{rcrcccrcccrcr} c_1\, x_1 & + & c_2\, x_2 & + & \ldots & + & c_n\, x_n & & & & & = & z \\ \hline a_{11}\, x_1 & + & a_{12}\, x_2 & + & \ldots & + & a_{1n}\, x_n & + & x_{n+1} & & & = & b_1 \\ a_{21}\, x_1 & + & a_{22}\, x_2 & + & \ldots & + & a_{2n}\, x_n & & & + \ x_{n+2} & & = & b_2 \\ \vdots & & \vdots & & & & \vdots & & & & & = & \vdots \\ a_{m1}\, x_1 & + & a_{m2}\, x_2 & + & \ldots & + & a_{mn}\, x_n & & & & + \ x_{n+m} & = & b_m. \end{array}$$

At any iteration of the simplex method, the variables $x_1, \ldots, x_{n+m}$ are classified into two groups as basic variables and non-basic variables. Let B be the set of basic variables and N

be the set of non-basic variables. We recall that there are m basic variables and n non-basic variables so that $|B| = m$ and $|N| = n$. Thus, the system of equations at any iteration of the simplex method has the form

$$\begin{aligned} \sum_{j \in N} \bar{c}_j \, x_j \qquad & = z - \alpha \\ \sum_{j \in N} \bar{a}_{ij} \, x_j + x_i = \bar{b}_i \qquad & \forall \, i \in B, \end{aligned}$$

where the first row above corresponds to the objective row and the remaining rows correspond to the constraint rows. The objective function coefficient of the non-basic variable x_j in the current system of equations is $\bar{c}_j$. There is one constraint row associated with each one of the basic variables $\{x_i : i \in B\}$. The non-basic variable x_j appears with a coefficient of $\bar{a}_{ij}$ in the constraint corresponding to the basic variable x_i. The right side of the constraint associated with the basic variable x_i is $\bar{b}_i$. We can obtain a solution to the system of equations above by setting $x_i = \bar{b}_i$ for all $i \in B$, $x_j = 0$ for all $j \in N$ and $z = \alpha$.

If $\bar{c}_j \leq 0$ for all $j \in N$, then we stop. The solution corresponding to the current system of equations is optimal. Otherwise, we pick a non-basic variable $k \in N$ such that $k = \arg\max\{\bar{c}_j : j \in N\}$, which is the non-basic variable with the largest coefficient in the objective function row. We will increase the value of the non-basic variable x_k.

Consider each constraint row $i \in B$. The basic variable x_i is the decision variable that appears only in this row. If $\bar{a}_{ik} > 0$, then an increase in x_k can be made up for by a decrease in x_i. In particular, we can increase x_k up to $\bar{b}_i/\bar{a}_{ik}$, while making sure that x_i remains non-negative. If $\bar{a}_{ik} < 0$, then an increase in x_k can be made up for by an increase in x_i. Therefore, if we increase x_k, we do not run into the danger x_i going negative. If $\bar{a}_{ik} = 0$, then increasing x_k makes no change in constraint i. Thus, we can increase x_k up to

$$\min_{i \in B} \left\{ \frac{\bar{b}_i}{\bar{a}_{ik}} : \bar{a}_{ik} > 0 \right\},$$

while making sure that none of the other variables become negative and all of the constraints remain satisfied. If we increase x_k to the value above, then the new value of the decision variable x_k is determined by the constraint $\ell = \arg\min_{i \in B}\{\frac{\bar{b}_i}{\bar{a}_{ik}} : \bar{a}_{ik} > 0\}$. Thus, the entering variable is x_k and the leaving variable is x_ℓ. We carry out row operations such that the decision variable x_k appears with a coefficient of 1 only in constraint row ℓ.

Initial Feasible Solutions and Linear Programs in General Form

When applying the simplex method on the linear programs that we considered so far, we could find an initial feasible solution without too much difficulty. In this chapter, we see that there are linear programs where it may be difficult to find an initial feasible solution for the simplex method to start with. We give a structured approach to come up with an initial feasible solution for these linear programs. Also, all of the linear programs we considered so far involved maximizing an objective function with less than or equal to constraints and non-negative decision variables. We discuss how we can deal with more general linear programs that have other types of objective functions, constraints and decision variables. We will see that we if we can solve linear programs that involves maximizing an objective function with less than or equal to constraints and non-negative decision variables, then we can actually deal with much more general linear programs.

1 Basic Variables and Spotting an Initial Feasible Solution

Consider the linear program

$$\begin{aligned} \max \quad & 3\,x_1 + 2\,x_2 \\ \text{st} \quad & x_1 + 2\,x_2 \leq 12 \\ & 2\,x_1 + x_2 \leq 11 \\ & x_1 + x_2 \leq 7 \\ & x_1, x_2 \geq 0. \end{aligned}$$

Using the slack variables w_1, w_2 and w_3 associated with the three constraints above, this linear program is equivalent to

$$\begin{aligned} \max \quad & 3\,x_1 + 2\,x_2 \\ \text{st} \quad & x_1 + 2\,x_2 + w_1 = 12 \\ & 2\,x_1 + x_2 + w_2 = 11 \\ & x_1 + x_2 + w_3 = 7 \\ & x_1, x_2, w_1, w_2, w_3 \geq 0. \end{aligned}$$

In this case, the simplex method starts with the system of equations

$$\begin{array}{rcrcrcrcrcr} 3\,x_1 & + & 2\,x_2 & & & & & & & = & z \\ \hline x_1 & + & 2\,x_2 & + & w_1 & & & & & = & 12 \\ 2\,x_1 & + & x_2 & & & + & w_2 & & & = & 11 \\ x_1 & + & x_2 & & & & & + & w_3 & = & 7. \end{array}$$

Recall the following properties of basic variables. First, each basic variable appears in exactly one constraint row with a coefficient of one. Second, each basic variable appears in a different

coefficient row. Third, the basic variables do not appear in the objective function row. Due to these properties, it is simple to spot a solution that satisfies the system of equations that the simplex method visits.

In the system of equations above, the basic variables are w_1, w_2 and w_3, whereas the non-basic variables are x_1 and x_2. The solution corresponding to the system of equations above is

$$w_1 = 12,\ w_2 = 11,\ w_3 = 7,\ x_1 = 0,\ x_2 = 0.$$

Also, since the basic variables do not appear in the objective function row and the non-basic variables take the value 0, we can easily find the value of z that satisfies the system of equations above. In particular, we have $z = 0$.

The solution $(x_1, x_2, w_1, w_2, w_3) = (0, 0, 12, 11, 7)$ is feasible to our linear program. The simplex method starts with this feasible solution and visits other feasible solutions while improving the value of the objective function. In the linear program above, it was simple to find a feasible solution for the simplex method to start with. As we show in the next section, it may not always be easy to find an initial feasible solution. To deal with this difficulty, we develop a structured approach to find an initial feasible solution.

2 Looking for a Feasible Solution

Consider the linear program

$$\begin{aligned} \max \quad & x_1 + x_2 \\ \text{st} \quad & x_1 - 3\,x_2 \leq -28 \\ & x_2 \leq 20 \\ & -x_1 - x_2 \leq -24 \\ & x_1, x_2 \geq 0. \end{aligned}$$

If we associate slack variables w_1, w_2 and w_3 with the three constraints above, then this linear program is equivalent to

The simplex method starts with the system of equations

$$\begin{array}{rcrcrcrcrcr} x_1 & + & x_2 & & & & & & & = & z \\ \hline x_1 & - & 3\,x_2 & + & w_1 & & & & & = & -28 \\ & & x_2 & & & + & w_2 & & & = & 20 \\ -x_1 & - & x_2 & & & & & + & w_3 & = & -24. \end{array}$$

In the system of equations above, the basic variables are w_1, w_2 and w_3, whereas the non-basic variables are x_1 and x_2. For the system of equations above, we have the solution

$$w_1 = -28,\ w_2 = 20,\ w_3 = -24,\ x_1 = 0,\ x_2 = 0,\ z = 0.$$

This solution is not feasible for the linear program above. In fact, we do not even know that there exists a feasible solution to the linear program! So, we focus on the question of how we can find a feasible solution to the linear program and how we can use this solution as the initial solution for the simplex method.

Consider the linear program

We call this linear program as the phase-1 linear program since we will use this linear program to obtain an initial feasible solution for the simplex method. We call the decision variable u as the artificial decision variable. The phase-1 linear program always has a feasible solution since setting $u = 28$, $x_1 = 0$ and $x_2 = 0$ provides a feasible solution to it. In the objective function of the phase-1 linear program, we minimize u. So, if possible at all, at the optimal solution to the phase-1 linear program, we want to set the value of the decision variable u to 0. Observe that if $u = 0$ at the optimal solution to the phase-1 linear program, then the optimal values of the decision variables x_1 and x_2 satisfy

$$x_1 - 3\,x_2 \leq -28, \qquad x_2 \leq 20, \qquad -x_1 - x_2 \leq -24, \qquad x_1 \geq 0, \qquad x_2 \geq 0,$$

which implies that these values of the decision variables are feasible to the original linear program that we want to solve. Therefore, if we solve the phase-1 linear program and the value of the decision variable u is 0 at the optimal solution, then we can use the optimal values of the decision variables x_1 and x_2 as an initial feasible solution to the original linear program that we want to solve.

On the other hand, if we have $u > 0$ at the optimal solution to the phase-1 linear program, then it is not possible to set the value of the decision variable to u to 0 and still obtain a

feasible solution to the phase-1 linear program, which implies that there do not exist x_1 and x_2 that satisfy

$$x_1 - 3\,x_2 \leq -28, \qquad x_2 \leq 20, \qquad -x_1 - x_2 \leq -24, \qquad x_1 \geq 0, \qquad x_2 \geq 0.$$

Thus, if we have $u > 0$ at the optimal solution to the phase-1 linear program, then the original linear program that we want to solve does not have a feasible solution. In other words, the original linear program that we want to solve is not feasible.

This discussion shows that to obtain a feasible solution to the linear program that we want to solve, we can first solve the phase-1 linear program. If we have $u = 0$ at the optimal solution to the phase-1 linear program, then the values of the decision variables x_1 and x_2 provide a feasible solution to the original linear program that we want to solve. If we have $u > 0$ at the optimal solution to the phase-1 linear program, then there does not exist a feasible solution to the original linear program.

So, we proceed to solving the phase-1 linear program. Associating the slack variables w_1, w_2 and w_3 with the three constraints and moving the decision variable u to the left side of the constraints, the simplex method starts with the system of equations

In the system of equations above, the basic variables are w_1, w_2 and w_3, whereas the non-basic variables are x_1, x_2 and u. The solution corresponding to the system of equations above is given by

$$w_1 = -28,\ w_2 = 20,\ w_3 = -24,\ x_1 = 0,\ x_2 = 0,\ u = 0,\ z = 0.$$

Note that this solution is not feasible to the phase-1 linear program because $x_1 - 3\,x_2 = 0 > -28 = -28 + u$. However, with only one set of row operations on the system of equations above, we can immediately obtain an equivalent system of equations such that we can spot a feasible solution to the phase-1 linear program from the new equivalent system of equations. In particular, we focus on the constraint row that has the most negative right side. We subtract this constraint row from every other constraint row and we add this constraint row to the objective function row. In particular, we focus on the first constraint row above. We subtract this constraint row from every other constraint row and add it to the objective function row. Also, we multiply the first constraint row by -1. In this case, we obtain the system of equations

Since a system of equations remains equivalent when we apply row operations on it, the last two systems of equations are equivalent to each other. In the system of equations above, the basic variables are u, w_2 and w_3, whereas the non-basic variables are $x_1,\ x_2$ and w_1. The solution corresponding to the system of equations above is

$$u = 28,\ w_2 = 48,\ w_3 = 4,\ x_1 = 0,\ x_2 = 0,\ w_1 = 0,\ z = 28.$$

This solution is feasible for the phase-1 linear program. Now, we can apply the simplex method as before to obtain an optimal solution to the phase-1 linear program.

Since we are minimizing the objective function in the phase-1 linear program, in the system of equations above, we pick the decision variable that has the largest negative objective function coefficient, which is x_2 with an objective function coefficient of -3. We increase the value of this decision variable. Applying the simplex method as before, we can increase x_2 up to $\min\{28/3, 48/4, 4/2\} = 2$, while making sure that all of the other decision variables remain non-negative. In this case, the new value of the decision variable x_2 is determined by the third constraint row. Thus, we carry out row operations such that x_2 appears only in the third constraint row with a coefficient of 1. In other words, the entering variable is x_2 and the leaving variable is w_1. Carrying out the appropriate row operations, we obtain the system of equations

In the system of equations above, the basic variables are $u,\ w_2$ and x_2, whereas the non-basic variables are $x_1,\ w_1$ and w_3. Noting the objective function row, we increase the value of x_1. We can increase x_1 up to $\min\{22/2, 40/3\} = 11$, while making sure that all of the other decision variables remain non-negative. In this case, the new value of the decision variable x_1 is determined by the first constraint row. Thus, we carry our row operations such that x_1 appears only in the first constraint row with a coefficient of 1. In other words, the entering variable is x_1 and the leaving variable is u. Through appropriate row operations, we obtain the system of equations

In this system of equations, all coefficients in the objective function are non-negative, which implies there are no variables that we can increase to further reduce the value of the objective function. Thus, we reached the optimal solution for the phase-1 linear program. The basic variables in the system of equations above are $x_1,\ w_2$ and x_2, whereas the non-basic variables are $w_1,\ w_3$ and u. The solution corresponding to the last system of equations is

$$x_1 = 11,\ w_2 = 7,\ x_2 = 13,\ w_1 = 0,\ w_3 = 0,\ u = 0,\ z = 0.$$

Since we have $u = 0$ at the optimal solution to the phase-1 linear program, we can use the values of the decision variables x_1 and x_2 as an initial feasible solution to the original linear program. In particular, $x_1 = 11$ and $x_2 = 13$ provides a feasible solution to the original linear program that we want to solve. We use this solution as an initial feasible solution when we use the simplex method to solve the original linear program.

3 Computing the Optimal Solution

By solving the phase-1 linear program in the previous section, we obtained a feasible solution to the original linear program that we want to solve. Now, we solve the original linear program starting from this feasible solution. The iterations of the simplex method in the previous section started with the system of equations

$$\begin{aligned} x_1 - 3\,x_2 + w_1 - u &= -28 \\ x_2 + w_2 - u &= 20 \\ -\,x_1 - x_2 + w_3 - u &= -24 \end{aligned}$$

for the constraints. After applying a sequence of row operations, we ended up with the system of equations

$$\begin{aligned} x_1 + \frac{1}{4}\,w_1 - \frac{3}{4}\,w_3 + \frac{1}{2}\,u &= 11 \\ \frac{1}{4}\,w_1 + w_2 + \frac{1}{4}\,w_3 - \frac{3}{2}\,u &= 7 \\ x_2 - \frac{1}{4}\,w_1 - \frac{1}{4}\,w_3 + \frac{1}{2}\,u &= 13. \end{aligned}$$

Putting aside the non-negativity constraints on the variables, the constraints of the original linear program that we want to solve are given by

$$\begin{aligned} x_1 - 3\,x_2 + w_1 &= -28 \\ x_2 + w_2 &= 20 \\ -\,x_1 - x_2 + w_3 &= -24. \end{aligned}$$

Thus, if we apply the same sequence of row operations that we applied in the previous section, then we would end up with the system of equations

$$\begin{aligned} x_1 + \frac{1}{4}\,w_1 - \frac{3}{4}\,w_3 &= 11 \\ \frac{1}{4}\,w_1 + w_2 + \frac{1}{4}\,w_3 &= 7 \\ x_2 - \frac{1}{4}\,w_1 - \frac{1}{4}\,w_3 &= 13. \end{aligned}$$

Since a system of equations remains equivalent after applying a sequence of row operations, this discussion implies that the last system of equations above are equivalent to the constraints of the original linear program. Thus, noting that we maximize $x_1 + x_2$ in the objective function of the original linear program, to solve the original linear program, we can start with the system of equations

In the system of equations above, we are tempted to identify x_1, w_2 and x_2 as the basic variables, but we observe that the decision variables x_1 and x_2 have non-zero coefficients in the objective function row, while the basic variables need to have a coefficient of 0 in the objective row. However, with only one set of row operations, we can immediately obtain a new system of equations that is equivalent to the one above and the decision variables x_1, w_2 and x_2 appear only in one of the constraints with a coefficient of 1 without appearing in the objective function row. In particular, we multiply the first constraint row by -1 and add it to the objective row. We multiply the third constraint row by -1 and add it to the objective row. Thus, we obtain the system of equations

Noting that a system of equations remains equivalent when we apply row operations on it, the last two system of equations are equivalent to each other. In the last system of equations above, we can now identify x_1, w_2 and x_2 as the basic variables, whereas w_1 and w_3 as the non-basic variables. For this system of equations, we have the solution

$$x_1 = 11,\ w_2 = 7,\ x_2 = 13,\ w_1 = 0,\ w_3 = 0,\ z = 24,$$

which is a feasible solution to the linear program that we want to solve and this solution provides an objective value of 24.

Since we are maximizing the objective function in our linear program, noting the objective function row in the system of equations above, we increase the value of the decision variable w_3. We can increase w_3 up to $7/\frac{1}{4} = 28$ while making sure that all of the other decision variables remain non-negative. In this case, the new value of the decision variable w_3 is determined by the second constraint row. Thus, we carry out row operations such that w_3 appears only in the second constraint row with a coefficient of 1. In other words, the entering variable is w_3 and the leaving variable is w_2. Applying the appropriate row operations, we obtain the system of equations

In this system of equations, x_1, w_3 and x_2 are the basic variables, whereas w_1 and w_3 are the non-basic variables. The solution corresponding to the system of equations above is

$$x_1 = 32,\ w_3 = 28,\ x_2 = 20,\ w_1 = 0,\ w_2 = 0,\ z = 52.$$

Since all of the objective function row coefficients are non-positive, we conclude that the solution $(x_1, x_2) = (32, 20)$ is an optimal solution providing an objective value of 52.

4 Linear Programs in General Form

All of the linear programs that we considered so far are of the form

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} c_j\, x_j & \\ \text{st} \quad & \sum_{j=1}^{n} a_{ij}\, x_j \leq b_i & \forall\, i = 1, \ldots, m \\ & x_j \geq 0 & \forall\, j = 1, \ldots, n. \end{aligned}$$

In particular, we maximize the objective function with less than or equal to constraints and non-negative decision variables. Not all linear programs have this form, but we can always bring a linear program into the form above, where we maximize the objective function with less than or equal to constraints and non-negative decision variables.

If we are minimizing the objective function $\sum_{j=1}^{n} c_j \, x_j$ in a linear program, then we can equivalently maximize the objective function $-\sum_{j=1}^{n} c_j \, x_j$.

If we have a greater than or equal to constraint of the form $\sum_{j=1}^{n} a_{ij} \, x_j \geq b_i$, then we can equivalently write this constraint as a less than or equal to constraint as $-\sum_{j=1}^{n} a_{ij} \, x_j \leq -b_i$.

If we have an equal to constraint of the form $\sum_{j=1}^{n} a_{ij} \, x_j = b_i$, then we can equivalently write this constraint as two inequality constraints $\sum_{j=1}^{n} a_{ij} \, x_j \leq b_i$ and $\sum_{j=1}^{n} a_{ij} \, x_j \geq b_i$.

If we have a decision variable x_j that takes non-positive values, then we can use a new decision variable y_j that takes non-negative values and replace all occurrences of x_j by $-y_j$.

Finally, if we have a non-restricted decision variable x_j that takes both positive and negative values, then we can use two new non-negative decision variables $\hat{x}_j$ and $\bar{x}_j$ to replace all occurrences of x_j by $\hat{x}_j - \bar{x}_j$. By using the transformations above, we can convert any linear program into a form where we maximize the objective function with less than or equal to constraints and non-negative decision variables. Consider the linear program

$$\begin{array}{rl} \min & 5\,x_1 - 9\,x_2 + 3\,x_3 + 4\,x_4 \\ \text{st} & x_1 + 7\,x_2 + 5\,x_3 + 2\,x_4 = 9 \\ & 2\,x_1 + 3\,x_3 \geq 7 \\ & x_2 + 6\,x_4 \leq 4 \\ & x_1 \geq 0, \; x_2 \text{ is free}, \; x_3 \leq 0, \; x_4 \geq 0. \end{array}$$

This linear program is equivalent to

which is, in turn, equivalent to

This discussion shows that it is enough to consider linear programs of the form where we maximize the objective function with less than or equal to constraints and non-negative decision variables because we can always convert any linear program into this form.

Unbounded Linear Programs, Multiple Optima and Degeneracy

There are linear programs where the objective function can be made arbitrarily large without violating the constraints. We refer to such linear programs as unbounded linear programs. Also, there are linear programs with multiple optimal solutions. In this chapter, we discuss how the simplex method can detect whether a linear program is bounded and whether a linear program has multiple optimal solutions. Furthermore, as the simplex method visits consecutive solutions, we may run into solutions where some basic variables take value 0. In such cases, the simplex method can carry out iterations without improving the objective function value. We refer to this situation as degeneracy.

1 Unbounded Linear Programs

For a linear program with a large number of decision variables and constraints, it may not be easy to see whether the linear program is unbounded. Fortunately, the simplex method can detect whether a linear program is unbounded. Consider the linear program

$$\begin{aligned} \max \quad & 4\,x_1 + 6\,x_2 - 3\,x_3 \\ \text{st} \quad & 2\,x_1 + x_2 - 2\,x_3 \leq 3 \\ & 3\,x_1 + 3\,x_2 - 2\,x_3 \leq 4 \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

The simplex method starts with the system of equations

$$\begin{array}{rcrcrcrcrcl} 4\,x_1 & + & 6\,x_2 & - & 3\,x_3 & & & & & = & z \\ \hline 2\,x_1 & + & x_2 & - & 2\,x_3 & + & w_1 & & & = & 3 \\ 3\,x_1 & + & 3\,x_2 & - & 2\,x_3 & & & + & w_2 & = & 4. \end{array}$$

The basic variables above are w_1 and w_2. The non-basic variables are x_1, x_2 and x_3. This system of equations has the corresponding solution $w_1 = 3,\ w_2 = 4,\ x_1 = 0,\ x_2 = 0,\ x_3 = 0,\ z = 0$. We choose to increase the value of the decision variable x_2, since x_2 has the largest positive coefficient in the objective function row. We can increase x_2 up to $\min\{3/1, 4/3\} = 4/3$, while making sure that all of the other decision variables remain non-negative. Thus, we carry out row operations so that x_2 appears only in the second constraint row with a coefficient of 1. These row operations provide the system of equations

In the system of equations above, the basic variables are w_1 and x_2. The non-basic variables are x_1, x_3 and w_2. This system of equations yields the solution $w_1 = 5/3,\ x_2 = 4/3,\ x_1 =$

$0,\ x_3 = 0,\ w_2 = 0,\ z = 8$. We increase the value of the decision variable x_3 since it has the largest positive coefficient in the objective function row.

Considering the first constraint row above, since the decision variable x_3 appears with a negative coefficient in this constraint row, if we increase x_3, then we can make up for the increase in x_3 by increasing w_1. Thus, we can increase x_3 as much as we want without running into the danger of w_1 going negative, which implies that the first constraint row does not impose any restrictions on how much we can increase the value of x_3. Similarly, considering the second constraint row above, if we increase x_3, then we can make up for the increase in x_3 by increasing x_2. So, we can increase x_3 as much as we want without running into the danger of x_2 going negative. This discussion shows that we can increase x_2 as much as we want without running into the danger of any of the other variables going negative. Also, the objective function row coefficient of x_2 in the last system of equations is positive, which implies that the increase in x_2 will make the value of the objective function larger. Therefore, we can make the objective function value as large as we want without violating the constraints. In other words, this linear program is unbounded.

The moral of this story is that if the system of equations at any iteration of the simplex method has a non-basic variable such that this non-basic variable has a positive coefficient in the objective function row and has a non-positive coefficient in all of the constraint rows, then the linear program is unbounded.

We note that it is difficult to see a priori that the linear program we want to solve is unbounded. However, the simplex method detects the unboundedness of the linear program during the course of its iterations. Once the simplex method detects that the linear program is unbounded, we can actually provide explanation for the unboundedness. For the linear program above, for some $t \geq 0$, consider the solution

$$x_3 = t, \quad w_1 = \frac{5}{3} + \frac{4}{3}\,t, \quad x_2 = \frac{4}{3} + \frac{2}{3}\,t, \quad x_1 = 0, \quad w_2 = 0.$$

For any $t \geq 0$, we have

$$\begin{aligned} 2\,x_1 + x_2 - 2\,x_3 &= \left(\frac{4}{3} + \frac{2}{3}\,t\right) - 2\,t = \frac{4}{3} - \frac{4}{3}\,t \leq 3 \\ 3\,x_1 + 3\,x_2 - 2\,x_3 &= 3\,\left(\frac{4}{3} + \frac{2}{3}\,t\right) - 2\,t = 4 \\ x_1 = 0, \quad x_2 &= \frac{4}{3} + \frac{2}{3}\,t \geq 0, \quad x_3 = t \geq 0. \end{aligned}$$

Therefore, the solution above is feasible to the linear program that we want to solve for any value of $t \geq 0$. Also, this solution provides an objective value of $4\,x_1 + 6\,x_2 - 3\,x_3 = 6\left(\frac{4}{3} + \frac{2}{3}\,t\right) - 3t = 8 + t$. If we choose t arbitrarily large, then the solution above is feasible to the linear program that we want to solve, but the objective value $8 + t$ provided by this solution is arbitrarily large. So, the linear program is unbounded.

2 Multiple Optima

Similar to boundedness, it is difficult to see whether a linear program with a large number of decision variables and constraints has multiple optimal solutions. Fortunately, the simplex method also allows us to see whether a linear program has multiple optimal solutions. Consider the linear program

$$\begin{aligned} \max \quad & 7\,x_1 + 12\,x_2 - 3\,x_3 \\ \text{st} \quad & 6\,x_1 + 8\,x_2 - 2\,x_3 \leq 1 \\ & -3\,x_1 - 3\,x_2 + x_3 \leq 2 \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

To solve the linear program above, the simplex method starts with the system of equations

$$\begin{array}{rcrcrcrcrcl} 7\,x_1 & + & 12\,x_2 & - & 3\,x_3 & & & & & = & z \\ \hline 6\,x_1 & + & 8\,x_2 & - & 2\,x_3 & + & w_1 & & & = & 1 \\ -3\,x_1 & - & 3\,x_2 & + & x_3 & & & + & w_2 & = & 2. \end{array}$$

The basic variables above are w_1 and w_2. The non-basic variables are x_1, x_2 and x_3. The solution correspondong to this system of equations is $w_1 = 1,\ w_2 = 2,\ x_1 = 0,\ x_2 = 0,\ x_3 = 0,\ z = 0$. Since x_2 has the largest positive coefficient in the objective function row, we choose to increase the value of the decision variable x_2. We can increase x_2 up to $1/8$, while making sure that all of the other decision variables remain non-negative. Thus, we carry out row operations so that x_2 appears only in the first constraint row with a coefficient of 1. Through these row operations, we obtain the system of equations

In the system of equations above, the basic variables are x_2 and w_2. The non-basic variables are x_1, x_3 and w_1. The solution corresponding to this system of equations is

$$x_2 = \frac{1}{8},\ w_2 = \frac{19}{8},\ x_1 = 0,\ x_3 = 0,\ w_1 = 0,\ z = \frac{3}{2}.$$

Since the objective function row coefficients of all variables are non-positive, increasing any of the variables does not improve the value of the objective function. Thus, the solution above is optimal and the optimal objective value of the linear program is $3/2$.

Now, the critical observation is that x_3 is a non-basic variable whose objective function row coefficient happened to be 0. If we increase the value of this decision variable, then the value of the objective function does not increase, but the value of the objective function does not decrease either! So, it is harmless to try to increase the decision variable x_3. Let

us go ahead and increase the value of the decision variable x_3. We can increase x_3 up to $\frac{19}{8}/\frac{1}{4} = 19/2$. Thus, we do row operations so that x_3 appears only in the second constraint row with a coefficient of 1. We obtain the system of equations

The basic variables are x_2 and x_3. The non-basic variables are $x_1,\ w_1$ and w_2. The solution corresponding to the system of equations above is

$$x_2 = \frac{5}{2},\ x_3 = \frac{19}{2},\ x_1 = 0,\ w_1 = 0,\ w_2 = 0,\ z = \frac{3}{2}.$$

In the last system of equations, the objective function row coefficients are non-positive. Thus, the solution above is also optimal for the linear program and it provides an objective value of 3/2. The two solutions that we obtained are quite different from each other, but they are both optimal for the linear program, providing an objective value of 3/2.

The moral of this story is that if the final system of equations in the simplex method includes a non-basic variable whose coefficient in the objective function row is 0, then we have multiple optimal solutions to the linear program.

3 Degeneracy

In the linear programs that we considered so far, the basic variables always took strictly positive values. However, it is possible that some basic variables take value 0. In such cases, we say that there is degeneracy in the current solution and the simplex method may have to carry out multiple iterations without improving the value of the objective function. Consider the linear program

$$\begin{aligned} \max \quad & 12\,x_1 + 6\,x_2 + 16\,x_3 \\ \text{st} \quad & x_1 - 4\,x_2 + 4\,x_3 \leq 2 \\ & x_1 + 2\,x_2 + 2\,x_3 \leq 1 \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

We start with the system of equations

$$\begin{array}{rcrcrcrcrcr} 12\,x_1 & + & 6\,x_2 & + & 16\,x_3 & & & & & = & z \\ \hline x_1 & - & 4\,x_2 & + & 4\,x_3 & + & w_1 & & & = & 2 \\ x_1 & + & 2\,x_2 & + & 2\,x_3 & & & + & w_2 & = & 1. \end{array}$$

The basic variables are w_1 and w_2. The non-basic variables are x_1, x_2 and x_3. The solution corresponding to the system of equations above is

$$w_1 = 2,\ w_1 = 1,\ x_1 = 0,\ x_2 = 0,\ x_3 = 0,\ z = 0.$$

We increase the value of the decision variable x_3. We can increase x_3 up to $\min\{2/4, 1/2\} = 2$. Note that there is a tie in the last minimum operator. To break the tie, we arbitrarily assume that the new value of the decision variable x_3 is dictated by the first constraint row. In this case, we carry out row operations so that x_3 appears only in the first constraint row with a coefficient of 1. Thus, we obtain the system of equations

The basic variables are x_3 and w_2. The non-basic variables are x_1, x_2 and w_1. The solution corresponding to the system of equations above is given by

$$x_3 = \frac{1}{2},\ w_2 = 0,\ x_1 = 0,\ x_2 = 0,\ w_1 = 0,\ z = 8.$$

In the solution above, the basic variable w_2 takes value 0. This solution provides an objective value of 8.

We increase the value of the decision variable x_2, whose objective function row coefficient is 22 in the system of equations above. We can increase x_2 up to $0/4 = 0$ while making sure that all of the other decision variables remain non-negative. So, we carry out row operations so that x_2 appears only in the second constraint row with a coefficient of 1. In this case, we get the system of equations

In this system of equations, the basic variables are x_3 and x_2, whereas the non-basic variables are $x_1,\ w_1$ and w_2. The solution corresponding to the system of equations above is

$$x_3 = \frac{1}{2},\ x_2 = 0,\ x_1 = 0,\ w_1 = 0,\ w_2 = 0,\ z = 8.$$

Now, the basic variable x_2 takes value 0. We observe that the values of the decision variables in the last two solutions we obtained are identical. Only the classification of the variables as basic and non-basic has changed. Furthermore, the last two solutions both provide an objective value of 8 for the linear program. Thus, this iteration of the simplex method did not improve the objective value for the linear program at all.

We increase the decision variable x_1. We can increase x_1 up to $\min\{\frac{1}{2}/\frac{3}{8}, 0/\frac{1}{8}\} = 0$. Thus, the new value of the decision variable x_1 is determined by the second constraint. In this case, we carry out row operations to make sure that x_1 appears only in the second constraint row with a coefficient of 1. We obtain the system of equations

The basic variables are x_3 and x_1, whereas the non-basic variables are $x_2,\ w_1$ and w_2. The solution corresponding to the system of equations above is

$$x_3 = \frac{1}{2},\ x_1 = 0,\ x_2 = 0,\ w_1 = 0,\ w_2 = 0,\ z = 8.$$

Now, the basic variable x_1 takes value 0. Again, the values of the decision variables in the last three solutions are identical. Only the classification of the variables as basic and non-basic has changed. All of these three solutions provide an objective value of 8 for the linear program. In all of the iterations of the simplex method, we have at least one strictly positive objective function row coefficient. Therefore, we cannot verify that we reached an optimal solution. However, as we carry out the iterations of the simplex method, we are not able to improve the value of the objective function either.

We do not give up. Scanning over the objective function row coefficients of the last system of equations, we decide to increase the decision variable w_1. We can increase w_1 up to $\frac{1}{2}/\frac{1}{2} = 1$. Thus, the new value of w_1 is determined by the first constraint row. We carry out row operations to make sure that w_1 appears only in the first constraint row with a coefficient of 1. These row operations yield the system of equations

The basic variables are w_1 and x_1, whereas the non-basic variables are $x_2,\ x_3$ and w_2. The solution corresponding to the system of equations above is

$$w_1 = 1,\ x_1 = 1,\ x_2 = 0,\ x_3 = 0,\ w_2 = 0,\ z = 12.$$

The objective value provided by the solution above is 12. So, we finally obtained a solution that improves the value of the objective function from 8 to 12. In the last system of equations, since the objective function row coefficients of all of the decision variables are non-positive, the solution above is optimal for the linear program. We can stop.

The moral of this story is that we can have basic variables that take value 0. When we have basic variables that take value 0, we say that the current solution is degenerate. If we encounter a degenerate solution, then the simplex method may have to carry out multiple iterations without improving the value of the objective function.

Min-Cost Network Flow Problem

In this chapter, we study linear programs with an underlying network structure. Such linear programs become particularly useful in routing, logistics and matching applications. Along with formulating a variety of linear programs with numerous application areas, we discuss the properties of the optimal solutions to these linear programs.

1 Min-Cost Network Flow Problem

Consider the figure below depicting a network over which we transport a certain product. At nodes 1 and 2, we have 5 and 2 units of supply for the product. At nodes 4 and 5, we have 3 and 4 units of demand for the product. We do not have any supply or demand at node 3, but we can use this node as a transshipment point. The directed arcs represent the links over which we can transport the product. For example, we can transport the product from node 3 to node 2, but we cannot transport from node 2 to node 3. We use the set $\{(1,2),(1,3),(2,4),(3,2),(3,5),(4,5),(5,4)\}$ to denote the set of arcs in the network. If we transport one unit of product over arc (i,j), then we incur a shipment cost of c_{ij}. These unit shipment costs are indicated on each arc in the figure below. We want to figure out how to ship the product from the supply nodes to the demand nodes so that we incur the minimum shipment cost, while making sure that we do not violate the supply availabilities at the supply nodes and satisfy the demands at the demand nodes. Note that the total supply in the network is equal to the total demand. Thus, to satisfy the demand at the demand nodes, all of the supply at the supply nodes must be shipped out.

To formulate this problem as a linear program, we use the decision variable x_{ij} to capture the number of units that we ship over arc (i,j). Thus, our decision variables are $x_{12}, x_{13}, x_{24}, x_{32}, x_{35}, x_{45}$ and x_{54}. To understand how we can set up the constraints in our linear program, the figure below shows one possible feasible solution to the problem. The labels on the arcs show the number of units shipped on each arc. The arcs that do not have any flow of product on them are indicated in dotted lines. In particular, for the solution in the figure below, the values of the decision variables are

$$x_{12} = 0, \quad x_{13} = 5, \quad x_{24} = 6, \quad x_{32} = 4, \quad x_{35} = 1, \quad x_{45} = 3, \quad x_{54} = 0.$$

Concentrating on the supply node 2 with a supply of 2 units, this node receives 4 units from node 3. Also, counting the 2 units of supply at node 2, node 2 has now 6 units of product. So, the flow out of node 2 in the feasible solution is 6. Therefore, the flow in and out of a supply node i in a feasible solution must satisfy

$$\text{Total Flow into Node } i + \text{Supply at Node } i = \text{Total Flow out of Node } i,$$

which can equivalently be written as

$$\text{Total Flow out of Node } i - \text{Total Flow into Node } i = \text{Supply at Node } i.$$

On the other hand, concentrating on the demand node 4 with a demand of 3 units, this node receives 6 units from node 2. Out of these 6 units, 3 of them are used to serve the demand at node 4 and the remaining 3 become the flow out of node 4. Thus, the flow in and out of a demand node i in a feasible solution must satisfy

$$\text{Total Flow into Node } i = \text{Demand at Node } i + \text{Total Flow out of Node } i,$$

which can equivalently be written as

$$\text{Total Flow into Node } i - \text{Total Flow out of Node } i = \text{Demand at Node } i.$$

Node 3 is neither a demand node or a supply node. For such a node, the total flow out of the node must be equal to the total flow into the node. Thus, the linear programming formulation of the problem is given by

The problem above is called the min-cost network flow problem. It is common to call the constraints as the flow balance constraints. The first two constraints are the flow balance constraints for nodes 1 and 2, which are supply nodes. In these constraints, we follow the convention that (total flow out) − (total flow in) = (supply of the node). The last two constraints are the flow balance constraints for nodes 4 and 5, which are demand nodes. In these constraints, we follow the convention that (total flow in) − (total flow out) = (demand of the node). The third constraint is the flow balance constraint for node 3, which is neither a supply nor a demand node. In this constraint, we follow the convention that (total flow out) − (total flow in) = 0. The formulation above is perfectly fine, but it requires us to remember two different types of constraints for supply and demand nodes. To avoid remembering two different types of constraints, we multiply the flow balance constraints for the demand nodes by −1 to get the equivalent linear program

Now, all of the constraints in this linear program are of the form

$$\text{Total Flow out of Node } i - \text{Total Flow into Node } i = \text{Availability at Node } i,$$

where availability is a positive number at supply nodes and a negative number at demand nodes. The last linear program avoids the necessity to remember two different forms of constraints for the supply and demand nodes. Our constraints always have the form (total flow out) − (total flow in) = (availability at the node). We only need to remember that availability is positive at supply nodes and negative at demand nodes.

An interesting observation for the min-cost network flow problem is that one of the constraints in the problem is always redundant. For example, assume that we have a solution that satisfies the first, second, fourth and fifth constraints. If we add the first, second, fourth and fifth constraints, then we obtain

$$\begin{array}{rcrcrcrcrcrcrcr}
x_{12} & + & x_{13} & & & & & & & & & & = & 5 \\
-x_{12} & & & + & x_{24} & - & x_{32} & & & & & & = & 2 \\
 & & & - & x_{24} & & & & & + & x_{45} & - & x_{54} & = & -3 \\
 & & & & & & & - & x_{35} & - & x_{45} & + & x_{54} & = & -4 \\
\hline
 & & x_{13} & & & - & x_{32} & - & x_{35} & & & & & = & 0,
\end{array}$$

which is identical to the third constraint. Thus, if we have a solution that satisfies the first, second, fourth and fifth constraints, then it must automatically satisfy the third constraint. We do not need to explicitly impose the third constraint. Similarly, we can check that if we leave any one of the constraints out and add the four remaining constraints in the min-cost network flow problem, then we obtain the constraint that is left out. Thus, we can always omit one of the constraints without changing the optimal solution.

2 Integrality of the Optimal Solution

An important property of the min-cost network flow problem is that if all of the demand and supply quantities are integers, then there exists an optimal solution where all of the decision variables take on integer values. This property can be quite useful in practice. For example, if we are shipping cars, then we can be sure that when we solve the min-cost network flow problem, we obtain a solution where do not ship half a car to one location and half a car to another, even though we do not explicitly impose the integrality requirement in our formulation of the min-cost network flow problem.

The integrality of the optimal solution originates from the fact that when we apply the simplex method on the min-cost network flow problem, we never have to carry out a division operation and all multiplication operations we have to carry out are multiplications by -1. To intuitively see this phenomenon, consider the system of equations corresponding to the constraints of our min-cost network flow problem. Recalling that one of the constraints is redundant, we omit the third constraint, in which case, the system of equations corresponding the constraints of our min-cost network flow problem is

$$\begin{array}{rcrcrcrcrcrcr}
x_{12} & + & x_{13} & & & & & & & & & = & 5 \\
-x_{12} & & & + & x_{24} & - & x_{32} & & & & & = & 2 \\
& & & - & x_{24} & & & & + & x_{45} & - & x_{54} & = & -3 \\
& & & & & & - & x_{35} & - & x_{45} & + & x_{54} & = & -4.
\end{array}$$

We know that in a system of equations with four constraints, we have four basic variables. Assume that we use the simplex method to solve the min-cost network flow problem. We want to answer the question of what the system of equations for the constraints would look like when the basic variables are, for example, x_{13}, x_{24}, x_{32} and x_{45}. To answer this question, we carry out row operations in the system of equations above to make sure that x_{13}, x_{24}, x_{32} and x_{45} appear in a different constraint with coefficients of 1. The variable x_{13} already appears in the first constraint with a coefficient of 1 and nowhere else. We multiply the second constraint by -1 to get

$$\begin{array}{rcrcrcrcrcrcr}
x_{12} & + & x_{13} & & & & & & & & & = & 5 \\
x_{12} & & & - & x_{24} & + & x_{32} & & & & & = & -2 \\
& & & - & x_{24} & & & & + & x_{45} & - & x_{54} & = & -3 \\
& & & & & & - & x_{35} & - & x_{45} & + & x_{54} & = & -4,
\end{array}$$

so that x_{32} appears only in the second constraint with a coefficient of 1 and nowhere else. We subtract the third constraint from the second constraint and multiply the third constraint by -1 to get

$$\begin{array}{rcrcrcrcrcrcr} x_{12} & + & x_{13} & & & & & & & & & = & 5 \\ x_{12} & & & & + & x_{32} & & & - & x_{45} & + & x_{54} & = & 1 \\ & & & x_{24} & & & & & - & x_{45} & + & x_{54} & = & 3 \\ & & & & & & - & x_{35} & - & x_{45} & + & x_{54} & = & -4. \end{array}$$

Thus, x_{24} appears only in the third constraint with a coefficient of 1 and nowhere else. Finally, we subtract the fourth constraint from the second and third constraints, and multiply the fourth constraint by -1 to get

$$\begin{array}{rcrcrcrcrcrcr} x_{12} & + & x_{13} & & & & & & & & & = & 5 \\ x_{12} & & & & + & x_{32} & + & x_{35} & & & & = & 5 \\ & & & x_{24} & & & + & x_{35} & & & & = & 7 \\ & & & & & & & x_{35} & + & x_{45} & - & x_{54} & = & 4. \end{array}$$

So, x_{45} now appears in the fourth constraint only with a coefficient of 1. Thus, if the simplex method visited the solution with basic variables x_{13}, x_{24}, x_{32} and x_{45}, then the values of these decision variables would be $x_{13} = 5$, $x_{24} = 7$, $x_{32} = 5$ and $x_{45} = 4$. Note that we did not have to carry out a division operation to obtain these values. Also, all of the multiplication operations were multiplication by -1. As a result, the values of the decision variables x_{13}, x_{24}, x_{32} and x_{45} are obtained by adding and subtracting the supply and demand quantities in the original min-cost network flow problem. If the supply and demand quantities are integers, then the values of x_{13}, x_{24}, x_{32} and x_{45} are integers as well.

As another example, let us check what the system of equations for the constraints in the simplex method would look like when the basic variables are x_{12}, x_{13}, x_{24} and x_{35}. We start from the last system of equations above. Since this system of equations was obtained from the original constraints of the min-cost network flow problem by using row operations, this system of equations is equivalent to the original constraints of the min-cost network flow problem. The variable x_{13} appears only in the first constraint only with a coefficient of 1. To make sure that x_{12} appears only in the second constraint with a coefficient of 1, we subtract the second constraint from the first constraint to obtain

$$\begin{array}{rcrcrcrcrcrcr} & x_{13} & & - & x_{32} & - & x_{35} & & & & & = & 0 \\ x_{12} & & & + & x_{32} & + & x_{35} & & & & & = & 5 \\ & & x_{24} & & & + & x_{35} & & & & & = & 7 \\ & & & & & & x_{35} & + & x_{45} & - & x_{54} & = & 4. \end{array}$$

The variable x_{24} already appears only in the third constraint only with a coefficient of 1. To make sure that x_{35} appears only in the fourth constraint with a coefficient of 1, we add the fourth constraint to the first constraint and subtract the fourth constraint from

the second and third constraints. In this case, we obtain

$$\begin{array}{rrrrrrrrrrrr}
 & x_{13} & & - & x_{32} & & + & x_{45} & - & x_{54} & = & 4 \\
x_{12} & & & + & x_{32} & & - & x_{45} & + & x_{54} & = & 1 \\
 & & x_{24} & & & & - & x_{45} & + & x_{54} & = & 3 \\
 & & & & & x_{35} & + & x_{45} & - & x_{54} & = & 4.
\end{array}$$

Thus, if the simplex method visited the solution with basic variables x_{12}, x_{13}, x_{24} and x_{35}, then the values of these decision variables would be $x_{12} = 1$, $x_{13} = 4$, $x_{24} = 3$ and $x_{35} = 4$. Again, we only used addition and subtraction to obtain these values. In particular, we did not use any division operation.

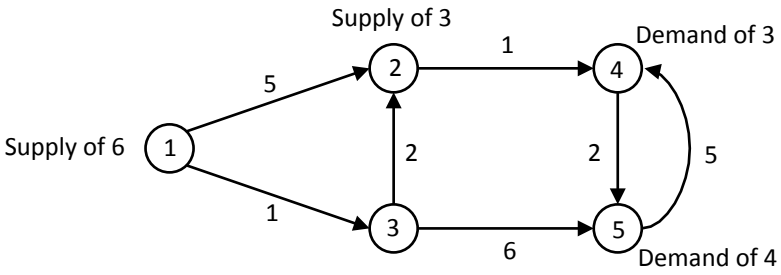
Although this discussion is not a theoretical proof, it convinces us that when we apply the simplex method on the min-cost network flow problem, we never have to use division and the only multiplication operation we use is multiplication by -1. So, the values of the decision variables in any solution visited by the simplex method are obtained by adding and subtracting the supply and demand quantities in the original problem. Thus, as long as the supply and demand quantities in the original problem take integer values, the decision variables will also take integer values in any solution visited by the simplex method. Since this observation applies to the final solution visited by the simplex method, the optimal solution to the min-cost network flow problem will be integer valued.

3 Min-Cost Network Flow Problem in Compact Form

To formulate the min-cost network flow problem in compact form, we first describe the data in compact form. We use N to denote the set of nodes and V to denote the set of arcs. We let S_i be the product availability at node i. Note that S_i is a positive quantity when node i is a supply node, but a negative quantity when node i is a demand node. We let C_{ij} be the cost of shipping a unit of product on arc (i, j). Thus, the data for the problem are $\{S_i : i \in N\}$ and $\{C_{ij} : (i, j) \in V\}$. We use the decision variable x_{ij} to capture the number of units that we ship on arc (i, j). To compute the total flow out of node i, we look at every arc that originates at node i and add the flows on these arcs. Thus, the total flow out of node i is given by $\sum_{j \in N:(i,j) \in V} x_{ij}$. In the last expression, we compute a sum over all $j \in N$ such that (i, j) is a valid arc in our min-cost network flow problem. Similarly, to compute the total flow into node i, we look at every arc that terminates at node i and add the flows on these arcs. Thus, the total flow into node i is given by $\sum_{j \in N:(j,i) \in V} x_{ji}$. The min-cost network flow problem can be formulated as

We observe that the constraints in the problem above are of the form (total flow out) − (total flow in) = (availability at the node).

In all of our min-cost network flow problems, the total supply in the network is equal to the total demand. Thus, to satisfy the demand at the demand nodes, all of the supply at the supply nodes must be shipped out. In certain applications, the total supply in the network may exceed the total demand, in which case, we do not have to ship out all of the supply at the supply nodes to satisfy the demand. Consider the min-cost network flow problem that takes place over the network in the figure below. This problem has the same data as the earlier min-cost network flow problem, but the supplies at nodes 1 and 2 are now 6 and 3 units. Thus, the total supply is 9, whereas the total demand is 7.



We want to figure out how to ship the product from the supply nodes to the demand nodes so that we incur the minimum shipment cost, while making sure that we do not violate the supply availabilities at the supply nodes and satisfy the demands at the demand nodes, but we do not need to ship out all the supply from the supply nodes. So, the flow in and out of a supply node i in a feasible solution must satisfy

$$\text{Total Flow into Node } i + \text{Supply at Node } i \geq \text{Total Flow out of Node } i,$$

which can equivalently be written as

$$\text{Total Flow out of Node } i - \text{Total Flow into Node } i \leq \text{Supply at Node } i.$$

We only need to adjust our constraints for the supply nodes. The constraints for the other nodes do not change. Using the decision variable x_{ij} with the same interpretation as before, we can formulate the problem as the linear program

$$\begin{aligned}
\min \quad & 5\,x_{12} + x_{13} + x_{24} + 2\,x_{32} + 6\,x_{35} + 2\,x_{45} + 5\,x_{54} \\
\text{st} \quad & x_{12} + x_{13} \leq 6 \\
& x_{24} - x_{12} - x_{32} \leq 3 \\
& x_{32} + x_{35} - x_{13} = 0 \\
& x_{45} - x_{24} - x_{54} = -3 \\
& x_{54} - x_{35} - x_{45} = -4 \\
& x_{12}, x_{13}, x_{24}, x_{32}, x_{35}, x_{45}, x_{54} \geq 0.
\end{aligned}$$

Observe that if the total supply is not equal to the total demand, then we have some inequality constraints in the min-cost network flow problem. In this case, it is not possible to choose any four of the five constraints and add them up to obtain the left out constraint. In particular, if we add up a number of inequalities and equalities, then we end up with an inequality, but the left out constraint could be an equality constraint. Thus, if we have some inequality constraints in our min-cost network flow problem, then none of the constraints in the min-cost network flow problem are redundant.

Our observations in the previous section continue to hold for the version of the min-cost network flow problem where we have some inequality constraints. In particular, even when the total supply in the network is not equal to the total demand, if all of the demand and supply quantities are integers, then there exists an optimal solution where all of the decision variables take on integer values.

The moral of this story is that the min-cost flow problem is a special type of linear program, where we can obtain integer solutions for free without explicitly imposing integrality requirements on our decision variables. We emphasize that this property is delicate and hinges on the fact that the only constraints in the min-cost network flow problem are of the form (total flow out) $-$ (total flow in) $=$ (availability at a node). If we impose additional constraints in the min-cost flow problem, then we can lose the integrality property of the optimal solution.

Assignment, Shortest Path and Max-Flow Problems

In this chapter, we study assignment, shortest path and max-flow problems, which can be viewed as special cases of the min-cost network flow problem with important applications.

1 Assignment Problem

We have three technicians and three jobs. Not all technicians are suitable for all jobs. If we assign a certain technician to a certain job, then we generate a reward depending on the technician we use. The table below shows the reward from assigning each technician to each job. For example, if we assign technician 2 to job 1, then we generate a reward of 3. Each job needs exactly one technician and each technician can do at most one job. So, since the number of jobs is equal to the number of technicians, each technician must be assigned to exactly one job. We want to figure out how to assign the technicians to the jobs so that we maximize the total reward obtained from our assignment decisions.

Tech \ Job	1	2	3
1	2	4	5
2	3	6	8
3	8	4	9

This problem can be formulated as a special min-cost network flow problem. In the figure below, we put one node on the left side for each technician. Each one of these nodes is a supply node with a supply of 1 unit. We put one node on the right side for each job. Each one of these nodes is a demand node with a demand of 1 unit. There is an arc from each technician node to each job node. Assigning technicians to jobs is equivalent to shipping out the supplies from the technician nodes to satisfy the demand at the job nodes. If we ship the supply at technician node i to satisfy the demand at job node j, then we are assigning technician i to job j, in which case, we get the reward of assigning technician i to job j.

We use x_{ij} to capture the number of units flowing on arc (i, j) in the figure above. We can

figure out how to ship the supplies from the technician nodes to cover the demand at the job nodes to maximize the total reward by solving the linear program

In this linear program, we maximize the objective function, but maximizing the objective function is equivalent to minimizing the negative of this objective function. We kept all of the constraints of the form (total flow out) − (total flow in) = (availability at the node), which is the form we used when formulating min-cost network flow problems in the previous chapter. Thus, this linear program corresponds to the min-cost network flow problem for the network depicted in the figure above. Since we know that min-cost network flow problems have integer valued optimal solutions, we do not need to worry about the possibility of sending half a unit of flow from a technician to one job and half a unit to another job in the optimal solution. Thus, the optimal solution to the linear program above provides a valid assignment of the technicians to the jobs. We refer to the problem above as the assignment problem. It is common to multiply the last three constraints in the formulation above by -1 and write the assignment problem as

$$\begin{aligned}
\max \quad & 2\,x_{11} + 4\,x_{12} + 5\,x_{13} + 3\,x_{21} + 6\,x_{22} + 8\,x_{23} + 8\,x_{31} + 4\,x_{32} + 9\,x_{33} \\
\text{st} \quad & x_{11} + x_{12} + x_{13} = 1 \\
& x_{21} + x_{22} + x_{23} = 1 \\
& x_{31} + x_{32} + x_{33} = 1 \\
& x_{11} + x_{21} + x_{31} = 1 \\
& x_{12} + x_{22} + x_{32} = 1 \\
& x_{13} + x_{23} + x_{33} = 1 \\
& x_{ij} \geq 0 \qquad \forall\, i = 1, 2, 3,\ j = 1, 2, 3,
\end{aligned}$$

in which case, the last three constraints ensure that we have a total flow of 1 into the demand node corresponding to each job. So, each job gets one technician. The first three constraints

ensure that we have a total flow of 1 out of each supply node corresponding to each tech. So, each tech is assigned to one job.

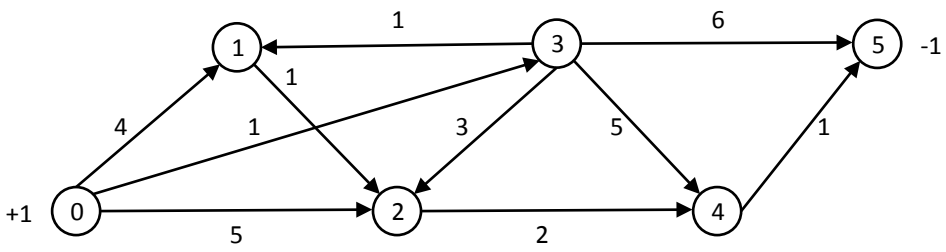
The optimal solution to the assignment problem above is given by $x_{12}^* = 1$, $x_{23}^* = 1$, $x_{31}^* = 1$. The other decision variables are 0 in the optimal solution. Therefore, an optimal solution to the assignment problem is obtained by assigning technician 1 to job 2, technician 2 to job 3 and technician 3 to job 1 with the corresponding optimal reward of 20.

To give a compact formulation of the assignment problem, assume that there are n technicians and n jobs. We let C_{ij} be the reward from assigning technician i to job j. We use the decision variable x_{ij} to capture the flow from the supply node corresponding to technician i to the demand node corresponding to job j. Observe that the total flow out of the supply node corresponding to technician i is $\sum_{j=1}^{n} x_{ij}$. Similarly, the total flow into the demand node corresponding to job j is $\sum_{i=1}^{n} x_{ij}$. Thus, the compact formulation of the assignment problem is

The moral of this story is that we can find the optimal assignment of technicians to jobs by a linear program without explicitly imposing the constraint that the assignment decisions should take integer values. This result follows from the fact that the assignment problem can be formulated as a min-cost network flow problem.

2 Shortest Path Problem

Consider the network in the figure below. We want to go from node 0 to node 5 by moving over the arcs. Each time we use an arc, we incur the cost indicated on the arc. For example, as we go from node 0 to node 5, if we use the arc $(3, 4)$, then we incur a cost of 5. We want to figure out how to go from node 0 to node 5 so that the total cost of the movement is minimized. In other words, we want to find the shortest path from node 0 to node 5, where the length of the path is the sum of the costs of the arcs in the path.



This problem can also be formulated as a special min-cost network flow problem. In the figure above, we put 1 unit of supply at the origin node 0 and 1 unit of demand at the destination node 5. If we ship a unit of flow over an arc, then we incur the cost indicated on the arc. Consider the problem of shipping the unit of supply at node 0 to satisfy the demand at node 5 while minimizing the cost of the shipment. This unit supply will travel over the path with the total minimum cost from node 0 to node 5. So, this unit will travel over the shortest path from node 0 to node 5. Thus, figuring out how to ship the unit of supply from node 0 to node 5 in the cheapest possible manner is equivalent to finding the shortest path from node 0 to node 5. We use the decision variable x_{ij} to capture the flow on arc (i, j). The problem of finding the cheapest possible way to ship the unit of supply from node 0 to node 5 can be solved as the min-cost network flow problem

The problem above is called the shortest path problem. In the constraints, we follow the convention that (total flow out) − (total flow in) = (availability at the node). The optimal solution to the problem above is given by $x^*_{03} = 1$, $x^*_{31} = 1$, $x^*_{12} = 1$, $x^*_{24} = 1$, $x^*_{45} = 1$. The other decision variables are 0 in the optimal solution. Thus, to go from node 0 to node 5 with the smallest possible cost, we go from node 0 to 3, from node 3 to node 1, from node 1 to node 2, from node 2 to node 4 and from node 4 to node 5.

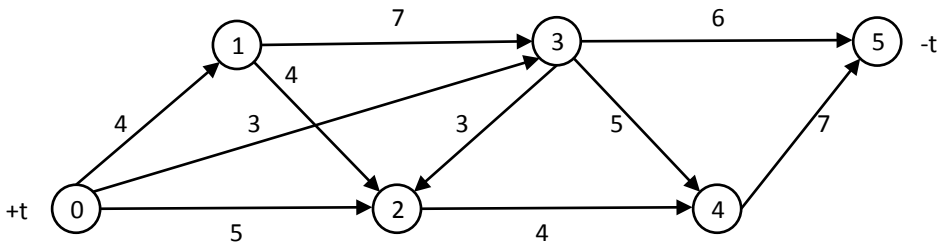
To give a compact formulation of the shortest path problem, we use $N = \{0, 1, \ldots, n\}$ to denote the set of nodes and A to denote the set of arcs in the network. We let 0 be the

origin node and n be the destination node. We use C_{ij} to denote the cost associated with moving over arc (i, j). The decision variable x_{ij} corresponds to the flow on arc (i, j). The compact formulation of the shortest path problem is

where the first and third constraints are the flow balance constraints for the origin and destination nodes, whereas the second constraint corresponds the flow balance constraints for all nodes other than the origin and destination nodes.

3 Max-Flow Problem

Consider the network in the figure below. The label on each arc gives the maximum flow allowed on each arc. For example, we allow at most 6 units of flow passing through arc $(3, 5)$. We want to figure out the maximum amount of flow we can push from node 0 to node 5, while adhering to the constraints on the maximum flow allowed on each arc. For example, we can push 8 units of flow from node 0 to node 5, where 4 units of flow follow the path through the nodes 0, 1, 3 and 5, whereas 4 units of flow follow the path through the nodes 0, 2, 4 and 5. Note that all of these flows satisfy the constraints on maximum flow allowed on each arc. Is 8 units the best we can do?



We can find the maximum amount of flow we can push from node 0 to node 5 by using a special min-cost network flow problem. In the network above, we put t units of supply at

node 0 and t units of demand at node 5, where t is a decision variable. If we can ship the t units of supply from node 0 to node 5 while adhering to the maximum flow allowed on each arc, then we can push t units of flow from node 0 to node 5. So, in our min-cost network flow problem, we maximize t, while making sure that the flows on the arcs satisfy the flow balance constraints and we adhere to the maximum flow allowed on each arc. Thus, using x_{ij} to denote the flow on arc (i, j), we solve the problem

This problem is called the max-flow problem. We emphasize that t is a decision variable in the problem above. The first constraint is the flow balance constraint for node 0. The sixth constraint is the flow balance constraint for node 5. The second to fifth constraints are the flow balance constraints for the nodes other than nodes 0 and 5. The last set of constraints ensures that the flows on the arcs adhere to the maximum flow allowed on each arc.

The optimal solution to the problem above is given by $t^* = 11,\ x^*_{01} = 4,\ x^*_{02} = 4,\ x^*_{03} = 3,\ x^*_{13} = 4,\ x^*_{24} = 4,\ x^*_{34} = 1,\ x^*_{35} = 6, x^*_{45} = 5$. The other decision variables are 0 in the optimal solution. Since $t^* = 11$, the maximum amount of flow we can push from node 0 to node 5 is 11 units.

To give a compact formulation of the max-flow problem, we use $N = \{0, 1, \ldots, n\}$ to denote the set of nodes and A to denote the set of arcs in the network. We use U_{ij} to denote the maximum flow allowed on arc (i, j). We want to find the maximum flow we can push from node 0 to node n. We use the decision variable x_{ij} to capture the flow on arc (i, j)

and the decision variable t to capture the flow that we push from node 0 to node n. The compact formulation of the max-flow problem is given by

The first and third constraints are the flow balance constraints for nodes 0 and n. The second set of constraints corresponds to the flow balance constraints for the nodes other than nodes 0 and n. The fourth set of constraints ensures that the flows on the arcs do not exceed the maximum flow allowed on each arc.

Using Gurobi to Solve Linear Programs

Gurobi is perhaps the strongest commercial linear programming package. When compared with building and solving linear programs with AMPL, the advantage of using Gurobi is that we can call Gurobi within a Python, Java or C++ program. For example, if we are developing an application that finds the shortest path between any origin and destination locations chosen by a user over a map, then we can develop the user interface by using Python, Java or C++. After the user choses the origin and destination locations in the application, we can call Gurobi within our application to solve the corresponding shortest path problem. Once Gurobi solves the shortest path problem, we can import the solution into our application and display the solution in the user interface. In this chapter, we discuss how to use Gurobi along with Python. By using approaches similar to the one discussed in this chapter, we can use Gurobi along with Java or C++ as well. Another strong linear programming package is CPLEX. The principles of working with CPLEX and Gurobi are essentially identical. Thus, we only go over Gurobi.

1 Gurobi as a Standalone Solver

The website for Gurobi is at `gurobi.com`. Gurobi is free for academic users. To obtain Gurobi, go to `http://user.gurobi.com/download/gurobi-optimizer` and make sure to register as an academic user. After registering, download and install the most recent version of Gurobi. Once Gurobi is installed, we need to activate the software license. Go to `http://user.gurobi.com/download/licenses/free-academic` and click on `Request License`. This action provides a license key number. To activate your software license, open a terminal window and type `grbgetkey` followed by the license key number. When Gurobi asks where to store the activated software license, simply choose the default location. Now, we are ready to use Gurobi.

We can use Gurobi as a standalone linear programming solver by reading the linear program that we want to solve from a text file. This feature does not require calling Gurobi within a Python program and it is particularly useful for users who do not know how to program. To demonstrate how to use Gurobi as a standalone linear programming solver, consider the following problem. We sell cloud computing services to two classes of customers, memory-intensive and storage-intensive. Both customer classes are served through yearly contracts. Each memory-intensive customer takes up 100 GB of RAM and 200 GB of disk space, whereas each storage-intensive customer takes up 40 GB of RAM and 400 GB of disk space. From each yearly contract with a memory-intensive customer, we make \$2400. From each yearly contract with a storage-intensive customer, we make \$3200. We have 10000 GB of RAM and 60000 GB of disk space available to sign contracts with the two customer classes. For technical reasons, we do not want to get in a contract with more than 140 storage-intensive customers. We want to figure out how many yearly contracts to sign with customers from each class to maximize the yearly revenue. We use the decision variables x_m

and x_s to respectively denote the number of contracts we sign with memory-intensive and storage-intensive customers. The problem we want to solve can be formulated as the linear program

$$\begin{aligned} \max \quad & 2400\,x_m + 3200\,x_s \\ \text{st} \quad & 100\,x_m + 40\,x_s \leq 10000 \\ & 200\,x_m + 400\,x_s \leq 60000 \\ & x_s \leq 140 \\ & x_m, x_s \geq 0. \end{aligned}$$

To solve the linear program above by using Gurobi, we construct a text file with the following contents and save it in a file named `cloud.lp`.

```
Maximize
  2400 xm + 3200 xs
Subject To
  ramConst : 100 xm + 40 xs <= 10000
  stoConst : 200 xm + 400 xs <= 60000
Bounds
  xs <= 140
End
```

The section titled `Maximize` indicates that we are maximizing the objective function. We provide the formula for the objective function by using the decision variables. We do not need to declare the decision variables separately. The section titled `Subject To` defines the constraints. We name the first constraints `ramConst` and provide the formula for this constraint. We define the second constraint similarly. The section `Bounds` gives the upper bounds on our decision variables. The decision variable `xs` has an upper bound of 140. We could list the bound on the decision variable `xs` as another constraint under the section titled `Subject To`, but if we list the upper bounds on the decision variables under the section titled `Bounds`, then Gurobi deals with the upper bounds more efficiently.

Once we have the text file that includes our linear programming model, we open a terminal window and type the command `gurobi.sh`, which runs Gurobi as a standalone linear programming solver. We can solve the linear program as follows.

```
gurobi> myModel = read("cloud.lp")
gurobi> myModel.optimize()
```

The command `myModel = read("cloud.lp")` reads the linear programming model in the file `cloud.lp` and stores this model to the variable `myModel`. If the file `cloud.lp` is not stored under the current working directory, then we need to provide the full path when reading the

file. The command `myModel.optimize()` solve sthe linear programming model stored in the variable `myModel`. In response to the two commands above, Gurobi displays the following output.

```
Optimize a model with 2 rows, 2 columns and 4 nonzeros
Coefficient statistics:
  Matrix range    [4e+01, 4e+02]
  Objective range [2e+03, 3e+03]
  Bounds range    [1e+02, 1e+02]
  RHS range       [1e+04, 6e+04]
Presolve time: 0.00s
Presolved: 2 rows, 2 columns, 4 nonzeros
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.4000000e+33   1.171875e+30   2.400000e+03      0s
       2    5.2000000e+05   0.000000e+00   0.000000e+00      0s
Solved in 2 iterations and 0.00 seconds
Optimal objective  5.200000000e+05
```

After solving the linear program, Gurobi informs us that the optimal objective value is 520,000. We can explore the optimal solution to the linear program as follows.

```
gurobi> myModel.printAttr("X")

    Variable            X
-------------------------
          xm           50
          xs          125
gurobi> myVars = myModel.getVars()
gurobi> print myVars
[<gurobi.Var xm (value 50.0)>, <gurobi.Var xs (value 125.0)>]
gurobi> print myVars[0].varName, myVars[0].x
xm 50.0
```

The command `myModel.printAttr("X")` prints the `"X"` attribute of the model stored in the variable `myModel`. This attribute includes the names and the values of the decision variables. The command `myVars = myModel.getVars()` stores the decision variables of the linear program in the array `myVars`. We can print this array by using the command `print myVars`. Note that printing the array `myVars` shows the names and the values of the decision variables. The command `print myVars[0].varName, myVars[0].x` prints the name and the value of the first decision variable. In particular, `myVars[0]` returns the first decision variable in the array `myVars[0]` and we access the name and the value of this decision variable by using the fields `varName` and `x`.

```
gurobi> myModel.printAttr("pi")

  Constraint           pi
-------------------------
    ramConst           10
    stoConst            7
gurobi> myConsts = myModel.getConstrs()
gurobi> print myConsts
[<gurobi.Constr ramConst>, <gurobi.Constr stoConst>]
gurobi> print myConsts[0].constrName, myConsts[0].pi
ramConst 10.0
```

In a following chapter, we will study duality theory. When we study duality theory, we will see that there is a dual variable associated with each constraint of a linear program. The command `myModel.printAttr("pi")` prints the `"pi"` attribute of our model. This attribute includes the names of the constraints and the values of the dual variables associated with the constraints. From the output above, the optimal value of the dual variable associated with the first constraint is 10. The command `myConsts = myModel.getConstrs()` stores the constraints in the array `myConsts`. We can print this array by using the command `print myConsts`. The output from printing the array `myConsts` is uninformative. It only shows the constraint names. The command `print myConsts[0].constrName, myConsts[0].pi` prints the name of the first constraint along with the value of the dual variable associated with this constraint. In particular, `myConsts[0]` returns the first constraint in the array `myConsts` and we access the name of this constraint and the optimal value of the dual variable associated with this constraint by using the fields `constrName` and `pi`.

We can use the following set of commands to open a file and write the names and the values of the decision variables into the file.

```
gurobi> outFile = open( "solution.txt", "w" )
gurobi> for curVar in myVars:
.......   outFile.write( curVar.varName + " " + str( curVar.x ) + "\n" )
.......
gurobi> outFile.close()
```

The command `outFile = open( "solution.txt", "w" )` opens the file `solutions.txt` for writing and assigns this file to the variable `outFile`. Recall that we stored the decision variables of our linear program in the array `myVars`. We use a for loop to go through all elements of this array and write the `varName` and `x` fields of each decision variable into the file. Lastly, we close the file. As may have been clear by now, interacting with Gurobi is similar to writing a Python script. Many other constructions that are available for writing Python scripts are also available when interacting with Gurobi.

2 Calling Gurobi within a Python Program

We continue using the linear program in the previous section to demonstrate how we can build and solve a linear program by calling Gurobi within a Python program. The following program builds and solves a linear program in Python.

```
from gurobipy import *

# create a new model
myModel = Model( "cloudExample" )

# create decision variables and integrate them into the model
xm = myModel.addVar( vtype = GRB.CONTINUOUS , name = "xm" )
xs = myModel.addVar( vtype = GRB.CONTINUOUS , name = "xs" , ub = 140 )
myModel.update()

# create a linear expression for the objective
objExpr = LinExpr()
objExpr += 2400 * xm
objExpr += 3200 * xs
myModel.setObjective( objExpr , GRB.MAXIMIZE )

# create expressions for constraints and add to the model
firstConst = LinExpr()
firstConst += 100 * xm
firstConst += 40 * xs
myModel.addConstr( lhs = firstConst , sense = GRB.LESS_EQUAL , \
                   rhs = 10000 , name = "ramConst" )
secondConst = LinExpr()
secondConst += 200 * xm
secondConst += 400 * xs
myModel.addConstr( lhs = secondConst , sense = GRB.LESS_EQUAL , \
                   rhs = 60000 , name = "stoConst" )

# integrate objective and constraints into the model
myModel.update()

# write the model in a file to make sure it is constructed correctly
myModel.write( filename = "testOutput.lp" )

# optimize the model
```

```
myModel.optimize()

# check the status of the model
curStatus = myModel.status
if curStatus in (GRB.Status.INF_OR_UNBD, GRB.Status.INFEASIBLE, \
              GRB.Status.UNBOUNDED):
    print( "Could not find the optimal solution" )
    exit(1)

# print optimal objective and optimal solution
print( "\nOptimal Objective: " + str( myModel.ObjVal ) )
print( "\nOptimal Solution:" )
myVars = myModel.getVars()
for curVar in myVars:
    print ( curVar.varName + " " + str( curVar.x ) )

# print optimal dual solution
print( "\nOptimal Dual Solution:" )
myConsts = myModel.getConstrs()
for curConst in myConsts:
    print ( curConst.constrName + " " + str( curConst.pi ) )
```

We start by creating a model and store our model in the variable `myModel`. Next, we create the decision variables and add them into our model. When creating a decision variable, we specify that the variable takes continuous values and give a name for the decision variable. If there is an upper or a lower bound on the decision variable, then we can specify these bounds as well. If we do not specify any upper and lower bounds, then the default choices are infinity for the upper bound and zero for the lower bound. Giving a name to the decision variable is optional. We store the two decision variables that we create in the variables `xm` and `xs`. The command `myModel.update()` is easy to overlook, but it is important. It ensures that our model `myModel` recognizes the variables `xm` and `xs`.

We proceed to creating the objective function and constraints of our model. Both the objective and the constraints are created by using the call `LinExpr()`, which creates an empty linear function. We construct the components of the linear function one by one. For the objective function, we create a linear function and store this linear function in the variable `objExpr`. Next, we indicate the coefficient of each decision variable in the objective function. Finally, we set the objective function of our model `myModel` to be the linear function `objExpr`. While doing so, we specify that we are maximizing the objective function. We create the constraints of our model somewhat similarly. For the first constraint, we create a linear function and store the linear function in the variable `firstConst`. Next, we indicate the coefficients of each decision variable in the constraint. Finally, we add the constraint to

our model `myModel`. When adding a constraint, the left side of the constraint is the linear function we created. The sense of the constraint can be `GRB.LESS_EQUAL`, `GRB.EQUAL` or `GRB.GREATER_EQUAL`. Giving a name to the constraint is optional. We deal with the second constraint similarly. Once we create and add the objective function and the constraints into the model, we use the call `myModel.update()` to ensure that our model recognizes the objective function and the constraints.

At this point, we created the full linear programming model. To make sure that nothing went wrong, we can write the model `myModel` into a file by using the call `myModel.write( filename = "testOutput.lp" )`. By inspecting the linear program that we write into the file `testOutput.lp`, we can make sure that we specified the objective function and the constraints correctly. Next, the call `myModel.optimize()` finds the optimal solution. By using `curStatus = myModel.status`, we store the current status of our model in the variable `curStatus`. If the status of our model corresponds to an infeasible or an unbounded solution, then we print a message and exit the program. In the remaining portion of the program, we access the optimal objective value, the optimal solution and the optimal dual solution and print these quantities. The approach that we use to access these quantities is identical to the approach that we followed when we used Gurobi as a standalone linear programming solver in the previous section.

3 Dealing with Large Linear Programs

When dealing with large linear programs, we use loops to create the decision variables and the constraints. In this section, we describe how we can use loops to create a linear programming model in Gurobi. For this purpose, we use the assignment problem that we studied in the previous chapter. Assume that we have three technicians and three jobs. If we assign a certain technician to a certain job, then we generate a reward depending on the technician we use. The table below gives the reward from assigning a certain technician to a certain job. We want to figure out how to assign the technicians to the jobs so that we maximize the total reward obtained from our assignment decisions.

Tech \ Job	1	2	3
1	2	4	5
2	3	6	8
3	8	4	9

We know that we can formulate this problem as the linear program

$$
\begin{aligned}
\max \quad & 2\,x_{11} + 4\,x_{12} + 5\,x_{13} + 3\,x_{21} + 6\,x_{22} + 8\,x_{23} + 8\,x_{31} + 4\,x_{32} + 9\,x_{33} \\
\text{st} \quad & x_{11} + x_{12} + x_{13} = 1 \\
& x_{21} + x_{22} + x_{23} = 1 \\
& x_{31} + x_{32} + x_{33} = 1 \\
& x_{11} + x_{21} + x_{31} = 1 \\
& x_{12} + x_{22} + x_{32} = 1 \\
& x_{13} + x_{23} + x_{33} = 1 \\
& x_{ij} \geq 0 \qquad \forall\, i = 1, 2, 3,\ j = 1, 2, 3,
\end{aligned}
$$

where the first three constraints ensure that each technician is assigned to one job and the last three constraints ensure that each job gets one technician. The problem above has nine decision variables and six constraints. In our Python program, we can certainly create nine decision variables and six constraints one by one, but this task would be tedious when the numbers of technicians and jobs get large. In the following Python program, we use loops to create the decision variables and the constraints. We present each portion of the program separately. We start by initializing the data for the problem.

```
from gurobipy import *

# there are 3 techs and 3 jobs
noTechs = 3
noJobs = 3

# initialize the reward data
rewards = [ [ 0 for i in range ( noTechs ) ] for j in range ( noJobs ) ]
rewards[ 0 ][ 0 ] = 2
rewards[ 0 ][ 1 ] = 4
rewards[ 0 ][ 2 ] = 5
rewards[ 1 ][ 0 ] = 3
rewards[ 1 ][ 1 ] = 6
rewards[ 1 ][ 2 ] = 8
rewards[ 2 ][ 0 ] = 8
rewards[ 2 ][ 1 ] = 4
rewards[ 2 ][ 2 ] = 9
```

The variables `noTechs` and `noJobs` keep the numbers of technicians and jobs. Since the numbers of technicians and jobs are equal to each other, there is really no reason to define two variables, but having two variables will be useful when we want to emphasize whether we are looping over the technicians or the jobs in the subsequent portions of our program. We use

the two-dimensional array `rewards` to store the reward values so that the (i, j)-th element of the array `rewards` includes the reward from assigning technician i to job j. In any reasonably large application, we would read the data for the problem from a file. To make our presentation clearer, we embedded the initialization of the data into our program, although this approach is not ideal when working on a large application.

Next, we create a new model and store it in the variable `myModel` and proceed to constructing the decision variables of our linear program.

```
# create a new model
myModel = Model( "assignmentExample" )

# create decision variables and store them in the array myVars
myVars = [ [ 0 for i in range ( noTechs ) ] for j in range ( noJobs ) ]
for i in range( noTechs ):
    for j in range ( noJobs ):
        curVar = myModel.addVar( vtype = GRB.CONTINUOUS , \
                                 name =  "x" + str( i ) + str( j ) )
        myVars[ i ][ j ] = curVar

# integrate decision variables into the model
myModel.update()
```

We have one decision variable for each technician and job pair. Each one of these decision variables takes continuous values. We name the decision variables by using the technician and job to corresponding to each decision variable. Lastly, we store all of the decision variables in the two-dimensional array `myVars`, so that the (i, j)-th element of the array `myVars` includes the decision variable corresponding to assigning technician i to job j. As in the previous section, by using the call `myModel.update()`, we make sure that our model `myModel` recognizes the decision variables we created.

After creating the decision variables in our linear program, we move on to defining the objective function as follows.

```
# create a linear expression for the objective
objExpr = LinExpr()
for i in range( noTechs ):
    for j in range ( noJobs ):
        curVar = myVars[ i ][ j ]
        objExpr += rewards[ i ][ j ] * curVar
myModel.setObjective( objExpr , GRB.MAXIMIZE )
```

The call `LinExp()` above creates a new linear function and we store this linear function in the variable `objExpr`. Recalling that there is one decision variable for each technician

and job pair, we loop over all technicians and jobs. By using the (i, j)-th element of the array `myVars`, we access the decision variable corresponding to assigning technician i to job j. We add this decision variable into the linear function for the objective function with the appropriate coefficient. Finally, we set the objective function of our model `myModel` to be the linear function `objExpr`.

Next, we create the constraints that ensure that each technician is assigned to one job. We need one of these constraints for each technician.

```
# create constraints so that each tech is assigned to one job
for i in range( noTechs ):
    constExpr = LinExpr()
    for j in range( noJobs ):
        curVar = myVars[ i ][ j ]
        constExpr += 1 * curVar
    myModel.addConstr( lhs = constExpr , sense = GRB.EQUAL , rhs = 1 , \
                       name = "t" + str( i ) )
```

We loop over all technicians. For technician i, we need to create a constraint that ensures that this technician is assigned to one job. We create a linear function that keeps the left side of this constraint and store this linear function in the variable `constExpr`. The decision variables that correspond to assigning technician i to any of the jobs appear in the constraint with a coefficient of 1. Thus, we loop over each job j and add the decision variable corresponding to assigning technician i to each job j into the constraint with a coefficient of 1. Now, we have a linear function corresponding to the left side of the constraint that ensures that technician i is assigned to one job. We add this constraint into our model as an equality constraint with a right side of 1. We name the constraint by using the technician corresponding to the constraint. By following the same approach, we create the constraints that ensure that each job gets one technician.

```
# create constraints so that each job gets one tech
for j in range( noJobs ):
    constExpr = LinExpr()
    for i in range( noTechs ):
        curVar = myVars[ i ][ j ]
        constExpr += 1 * curVar
    myModel.addConstr( lhs = constExpr , sense = GRB.EQUAL , rhs = 1 , \
                       name = "j" + str( i ) )

# integrate objective and constraints into the model
myModel.update()
```

After creating the objective function and the constraints, we use the call `myModel.update()`

to ensure that our model recognizes the objective function and the constraints that we created. Through the discussion so far, we fully built our linear program. In the remaining portion of the program, we write our linear program into a file to make sure that nothing went wrong, we solve our linear program and inspect the optimal solution. In the previous section, we already discussed how to write a linear program into a file, solve the linear program and inspect the optimal solution. So, the remaining portion of our program is borrowed from the program in the previous section.

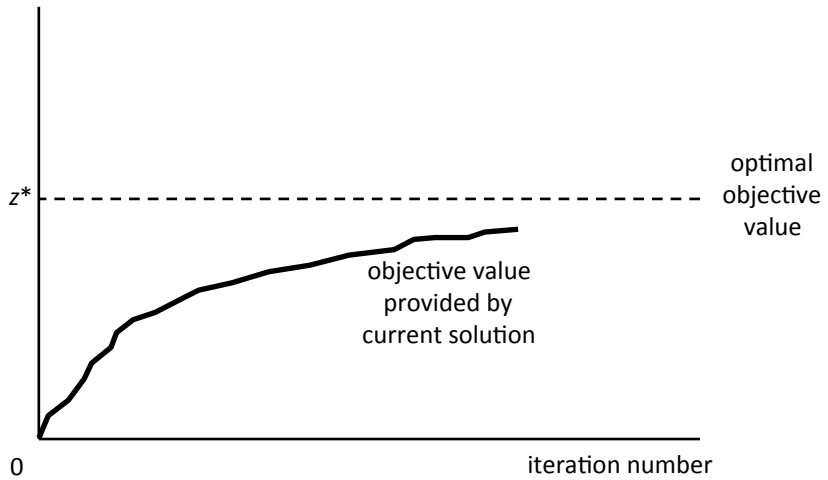
```
# write the model in a file to make sure it is constructed correctly
myModel.write( filename = "testOutput.lp" )

# optimize the model
myModel.optimize()

# print optimal objective and optimal solution
print( "\nOptimal Objective: " + str( myModel.ObjVal ) )
print( "\nOptimal Solution:" )
allVars = myModel.getVars()
for curVar in allVars:
    print ( curVar.varName + " " + str( curVar.x ) )
```

Introduction to Duality Theory and Weak Duality

Duality theory allows us to obtain upper bounds on the optimal objective value of a linear program. Such upper bounds become useful when it is computationally-intensive to obtain the optimal solution to a linear program and we stop the simplex method with a feasible, but not necessarily an optimal, solution. In such a case, we can compare the objective value provided by the feasible solution with the upper bound on the optimal objective value to get a feel for the optimality gap of the feasible solution on hand. In this chapter, we discuss how we can use duality theory to obtain an upper bound on the optimal objective value of a linear program and how we can use such an upper bound to understand the optimality gap of a feasible solution that we have on hand.

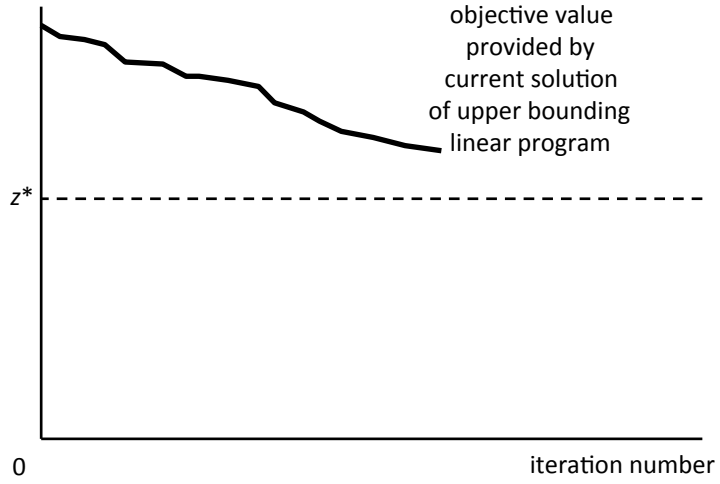
1 Motivation for Duality Theory

Assume that we want to solve a large-scale linear program by using the simplex method. As the iterations of the simplex method proceed, we obtain feasible solutions that provide larger and larger objective function values. Let us say the linear program is so large that we still have not obtained the optimal solution after two days of computation and we terminate the simplex method before it reaches an optimal solution. What we have on hand is a feasible solution, but we do not know how close this solution is to being optimal. Note that figuring out how close a feasible solution is to being optimal is not easy because we do not know the optimal objective value of the problem we want to solve. In the figure below, we depict the objective value provided by the solution on hand as a function of the iteration number in the simplex method. As the iterations progress, the objective value provided by the current solution gets larger and larger.



Imagine that we are able to construct another linear program such that this linear program is a minimization problem and the optimal objective value of this linear program is greater than or equal to the optimal objective of the original original linear we want to solve.

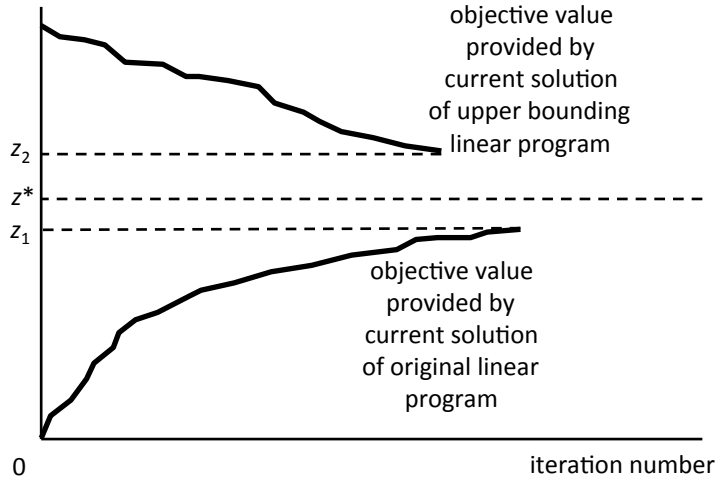
For the moment, we refer to this linear program as the upper bounding linear program. As we solve the original linear program we want to solve, we also solve the upper bounding linear program on another computer by using the simplex method. Since we minimize the objective function in the upper bounding linear program, as the iterations of the simplex method for the upper bounding linear program proceeds, we obtain feasible solutions for the upper bounding linear program that provides smaller and smaller objective function values. In the figure below, we depict the objective value provided by the solution on hand for the upper bounding linear program as a function of the iteration number in the simplex method. Since the optimal objective value of the upper bounding linear program is greater than or equal to the optimal objective value of the original linear program we want to solve, the objective value provided by the current solution in the figure below never dips below the optimal objective value of the original linear program we want to solve.



After two days of computation time, we terminate the simplex method for both the original linear program we want to solve and the upper bounding linear program. We want to understand how close the solution that we have for the original linear program is to being optimal. In the figure below, z_1 corresponds to the objective value provided by the solution that we have for the original linear program after two days of computation time. The percent optimality gap of this solution is $(z^* - z_1)/z^*$. We cannot compute this optimality gap because we do not know z^*. On the other hand, z_2 corresponds to the objective value provided by the solution that we have for the upper bounding linear program after two days of computation time. Note that we know z_2, which implies that we can compute $(z_2 - z_1)/z_1$. Furthermore, since the optimal objective value of the upper bounding linear program is greater than or equal to the optimal objective value of the original linear program we want to solve, we have $z_2 \geq z^* \geq z_1$. Thus, we obtain

$$\frac{z^* - z_1}{z^*} \leq \frac{z_2 - z_1}{z_1}.$$

The percent optimality gap of the solution that we have for the original linear program we want to solve is given by $(z^* - z_1)/z^*$, but we cannot compute this quantity. On the other hand, we can compute the quantity $(z_2 - z_1)/z_1$. Assume that $(z_2 - z_1)/z_1$ came out to be 1%. In this case, noting the inequality above, we can conclude that $(z^* - z_1)/z^* \leq 1\%$, which is to say that the optimality gap of the solution we have for the original linear program we want to solve is no larger than %1. In other words, the upper bounding linear program allows us to check the optimality gap of a solution that we have for the original linear program before we even obtain the optimal solution.



Motivated by the discussion above, the key question is how we can come up with an upper bounding linear program that satisfies two properties. First, the upper bounding linear program should be a minimization problem. Second, the optimal objective value of the upper bounding linear program should be an upper bound on the optimal objective value of the original linear program we want to solve.

2 Upper Bounds on the Optimal Objective Value

We want to solve the linear program

$$\begin{aligned} \max \quad & 5\,x_1 + 3\,x_2 - x_3 \\ \text{st} \quad & 3\,x_1 + 2\,x_2 + x_3 \leq 9 \\ & 4\,x_1 + x_2 - x_3 \leq 3 \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

For the sake of illustration, assume that this linear program is large enough that we cannot obtain its optimal objective value in reasonable computation time and we want to obtain an upper bound on its optimal objective value. Let (x_1^*, x_2^*, x_3^*) be the optimal solution to the linear program providing the optimal objective value $5\,x_1^* + 3\,x_2^* - x_3^*$. Since (x_1^*, x_2^*, x_3^*)

is the optimal solution to the linear program, it should satisfy the constraints of the linear program, so that we have

$$3\,x_1^* + 2\,x_2^* + x_3^* \le 9, \quad 4\,x_1^* + x_2^* - x_3^* \le 3, \quad x_1^* \ge 0, \quad x_2^* \ge 0, \quad x_3^* \ge 0.$$

We multiply the first constraint above by 1 and the second constraint above by 2 and add them up to obtain

Also, since $x_1^* \ge 0$, $x_2^* \ge 0$ and $x_3^* \ge 0$, we have $11\,x_1^* \ge 5\,x_1^*$, $4\,x_2^* \ge 3\,x_2^*$ and $-x_3^* \ge -x_3^*$. Adding these inequalities yield

Combining the two displayed inequalities above, we get $5\,x_1^* + 3\,x_2^* - x_3^* \le 11\,x_1^* + 4\,x_2^* - x_3^* \le 15$. Since the optimal objective value of the linear program is $5\,x_1^* + 3\,x_2^* - x_3^*$, the last inequality shows that 15 is an upper bound on the optimal objective value.

The key to the argument here is to combine the constraints by multiplying them with positive numbers in such a way that the coefficient of each variable in the combined constraint dominates its corresponding coefficient in the objective function.

A natural question is whether we can obtain an upper bound tighter than 15 by multiplying the two constraints with numbers other than 1 and 2. To answer this question, we generalize the idea by multiplying the constraints with generic numbers $y_1 \ge 0$ and $y_2 \ge 0$, instead of 1 and 2 As before, since (x_1^*, x_2^*, x_3^*) is the optimal solution to the linear program, it should satisfy the constraints of the linear program, so that we have

$$3\,x_1^* + 2\,x_2^* + x_3^* \le 9, \quad 4\,x_1^* + x_2^* - x_3^* \le 3, \quad x_1^* \ge 0, \quad x_2^* \ge 0, \quad x_3^* \ge 0.$$

We multiply the first constraint by $y_1 \ge 0$ and the second constraint by $y_2 \ge 0$ and add them up. Thus, if $y_1 \ge 0$ and $y_2 \ge 0$, then we have

Note that multiplying the constraints by positive y_1 and y_2 ensures we do not change the direction of the inequalities. Also, noting that $x_1^* \geq 0$, if $3\,y_1 + 4\,y_2 \geq 5$, then we have $(3\,y_1 + 4\,y_2)\,x_1^* \geq 5\,x_1^*$. Similarly, since $x_2^* \geq 0$, if $2\,y_1 + y_2 \geq 3$, then we have $(2\,y_1 + y_2)\,x_2^* \geq 3\,x_2^*$. Finally, since $x_3^* \geq 0$, if $y_1 - y_2 \geq -1$, then we have $(y_1 - y_2)\,x_3^* \geq -x_3^*$. Therefore, if $3\,y_1 + 4\,y_2 \geq 5$, $2\,y_1 + y_2 \geq 3$ and $y_1 - y_2 \geq -1$, then we have

Considering the last two displayed inequalities, the first one holds under the assumption that $y_1 \geq 0$ and $y_2 \geq 0$, whereas the second one holds under the assumption that $3\,y_1 + 4\,y_2 \geq 5$, $2\,y_1 + y_2 \geq 3$ and $y_1 - y_2 \geq -1$. Combining these two inequalities, it follows that if

$$y_1 \geq 0, \quad y_2 \geq 0, \quad 3\,y_1 + 4\,y_2 \geq 5, \quad 2\,y_1 + y_2 \geq 3, \quad y_1 - y_2 \geq -1,$$

then we have

$$5\,x_1^* + 3\,x_2^* - x_3^* \leq (3\,y_1 + 4\,y_2)\,x_1^* + (2\,y_1 + y_2)\,x_2^* + (y_1 - y_2)\,x_3^* \leq 9\,y_1 + 3\,y_2.$$

Thus, since the optimal objective value of the linear program we want to solve is $5\,x_1^* + 3\,x_2^* - x_3^*$, the last inequality above shows that $9\,y_1 + 3\,y_2$ is an upper bound on the optimal objective value of the linear program.

This discussion shows that as long as y_1 and y_2 satisfy the conditions $y_1 \geq 0$, $y_2 \geq 0$, $3\,y_1 + 4\,y_2 \geq 5$, $2\,y_1 + y_2 \geq 3$ and $y_1 - y_2 \geq -1$, the quantity $9\,y_1 + 3\,y_2$ is an upper bound on the optimal objective value of the linear program we want to solve. To obtain the tightest possible upper bound on the optimal objective value, we can push the upper bound $9\,y_1 + 3\,y_2$ as small as possible while making sure that the conditions imposed on y_1 and y_2 are satisfied. In other words, we can obtain the tightest possible upper bound on the optimal objective value by solving the linear program

$$\begin{aligned} \min \quad & 9\,y_1 + 3\,y_2 \\ \text{st} \quad & 3\,y_1 + 4\,y_2 \geq 5 \\ & 2\,y_1 + y_2 \geq 3 \\ & y_1 - y_2 \geq -1 \\ & y_1, y_2 \geq 0. \end{aligned}$$

The optimal objective value of the linear program above is an upper bound on the optimal objective value of the original linear program we want to solve. Furthermore, this linear program is a minimization problem. Therefore, we can use the linear program above as the upper bounding linear program as discussed in the previous section! In linear programming vocabulary, we refer to the upper bounding linear program above as the dual problem. We refer to the original linear program we want to solve as the primal problem.

3 Primal and Dual Problems

The primal and dual problems for the example in the previous section are

$$\begin{array}{lll} \max & 5\,x_1 + 3\,x_2 - x_3 & \\ \text{st} & 3\,x_1 + 2\,x_2 + x_3 \leq 9 & (y_1) \\ & 4\,x_1 + x_2 - x_3 \leq 3 & (y_2) \\ & x_1, x_2, x_3 \geq 0, & \end{array} \qquad \begin{array}{lll} \min & 9\,y_1 + 3\,y_2 & \\ \text{st} & 3\,y_1 + 4\,y_2 \geq 5 & (x_1) \\ & 2\,y_1 + y_2 \geq 3 & (x_2) \\ & y_1 - y_2 \geq -1 & (x_3) \\ & y_1, y_2 \geq 0. & \end{array}$$

By the discussion in the previous section, we can use the dual problem as the upper bounding linear program when we solve the primal problem. Note that for each constraint in the primal problem, we have a dual decision variable y_i. For each decision variable x_j in the primal problem, we have a constraint in the dual problem. The objective coefficient of dual variable y_i in the dual problem is the same as the right side of the primal constraint corresponding to variable y_i. The right side of dual constraint corresponding to variable x_j is the objective coefficient of primal variable x_j in the primal problem. The constraint coefficient of variable y_i in the dual constraint corresponding to variable x_j is the same as the constraint coefficient of variable x_j in the primal constraint corresponding to variable y_i.

Using the slack variables w_1 and w_2 for the primal constraints and the slack variables z_1, z_2 and z_3 for the dual constraints, we also can write the primal and dual problems as

$$\begin{array}{lll} \max & 5\,x_1 + 3\,x_2 - x_3 & \\ \text{st} & 3\,x_1 + 2\,x_2 + x_3 + w_1 = 9 & (y_1) \\ & 4\,x_1 + x_2 - x_3 + w_2 = 3 & (y_2) \\ & x_1, x_2, x_3, w_1, w_2 \geq 0, & \end{array} \qquad \begin{array}{lll} \min & 9\,y_1 + 3\,y_2 & \\ \text{st} & 3\,y_1 + 4\,y_2 - z_1 = 5 & (x_1) \\ & 2\,y_1 + y_2 - z_2 = 3 & (x_2) \\ & y_1 - y_2 - z_3 = -1 & (x_3) \\ & y_1, y_2, z_1, z_2, z_3 \geq 0. & \end{array}$$

Recall that for each constraint in the primal problem, we have a dual decision variable y_i. Also, for each constraint in the primal problem, we have a primal slack variable w_i. So, each dual decision variable y_i is associated with a primal slack variable w_i. On the other hand, for each decision variable x_j in the primal problem, we have a constraint in the dual problem. Also, for each constraint in the dual problem, we have a dual slack variable z_j. So, each primal decision variable x_j is associated with a dual slack variable z_j.

Another way to look at the relationship between the primal and dual problems is to write these problems in matrix notation. We define the matrices and the vectors

$$c = \begin{bmatrix} 5 \\ 3 \\ -1 \end{bmatrix} \qquad A = \begin{bmatrix} 3 & 2 & 1 \\ 4 & 1 & -1 \end{bmatrix} \qquad b = \begin{bmatrix} 9 \\ 3 \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}.$$

Using A^t to denote the transpose of a matrix A, the primal and dual problems considered in this section can be written in matrix notation as

$$\begin{array}{rl} \max & c^t\, x \\ \text{st} & A\, x \leq b \\ & x \geq 0, \end{array} \qquad\qquad \begin{array}{rl} \min & b^t\, y \\ \text{st} & A^t\, y \geq c \\ & y \geq 0. \end{array}$$

We can use the template above to write the dual problem corresponding to any general primal problem. Consider a primal problem in the general form

$$\begin{array}{rll} \max & \displaystyle\sum_{j=1}^{n} c_j\, x_j & \\ \text{st} & \displaystyle\sum_{j=1}^{n} a_{ij}\, x_j \leq b_i & \forall\, i = 1, \ldots, m \\ & x_j \geq 0 & \forall\, j = 1, \ldots, n. \end{array}$$

Defining the matrices and the vectors

$$c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} \quad A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix},$$

the primal problem above is of the form max $c^t\, x$ subject to $A\, x \leq b$ and $x \geq 0$, which implies that the dual problem corresponding to this primal problem has the form min $b^t\, y$ subject to $A^t\, y \geq c$ and $y \geq 0$. We can write the last problem equivalently as

$$\begin{array}{rll} \min & \displaystyle\sum_{i=1}^{m} b_i\, y_i & \\ \text{st} & \displaystyle\sum_{i=1}^{m} a_{ij}\, y_i \geq c_j & \forall\, j = 1, \ldots, n \\ & y_i \geq 0 & \forall\, i = 1, \ldots, m. \end{array}$$

Thus, in general form, a primal problem and its corresponding dual problem are given by

$$\begin{array}{rll} \max & \displaystyle\sum_{j=1}^{n} c_j\, x_j & \\ \text{st} & \displaystyle\sum_{j=1}^{n} a_{ij}\, x_j \leq b_i & \forall\, i = 1, \ldots, m \\ & x_j \geq 0 & \forall\, j = 1, \ldots, n, \end{array} \qquad\qquad \begin{array}{rll} \min & \displaystyle\sum_{i=1}^{m} b_i\, y_i & \\ \text{st} & \displaystyle\sum_{i=1}^{m} a_{ij}\, y_i \geq c_j & \forall\, j = 1, \ldots, n \\ & y_i \geq 0 & \forall\, i = 1, \ldots, m. \end{array}$$

4 Weak Duality

Weak duality states that the objective value of the dual problem at a feasible solution is at least as large as the objective value of the primal problem at a feasible solution. To see this relationship, consider the primal and dual problems

$$\begin{array}{rl} \max & 5\,x_1 + 3\,x_2 - x_3 \\ \text{st} & 3\,x_1 + 2\,x_2 + x_3 \leq 9 \\ & 4\,x_1 + x_2 - x_3 \leq 3 \\ & x_1, x_2, x_3 \geq 0, \end{array} \qquad \begin{array}{rl} \min & 9\,y_1 + 3\,y_2 \\ \text{st} & 3\,y_1 + 4\,y_2 \geq 5 \\ & 2\,y_1 + y_2 \geq 3 \\ & y_1 - y_2 \geq -1 \\ & y_1, y_2 \geq 0. \end{array}$$

Let $(\hat{x}_1, \hat{x}_2, \hat{x}_3)$ be a feasible solution to the primal problem and $(\hat{y}_1, \hat{y}_2)$ be a feasible solution to the dual problem. Since $(\hat{x}_1, \hat{x}_2, \hat{x}_3)$ is a feasible solution to the primal problem, we have $9 \geq 3\,\hat{x}_1 + 2\,\hat{x}_2 + \hat{x}_3$ and $3 \geq 4\,\hat{x}_1 + \hat{x}_2 - \hat{x}_3$. Also, $\hat{y}_1 \geq 0$ and $\hat{y}_2 \geq 0$, since $(\hat{y}_1, \hat{y}_2)$ is a feasible solution to the dual problem. Therefore, we have

$$\begin{array}{rrcl} & \hat{y}_1\,9 & \geq & \hat{y}_1\,(3\,\hat{x}_1 + 2\,\hat{x}_2 + \hat{x}_3) \\ + & \hat{y}_2\,3 & \geq & \hat{y}_2\,(4\,\hat{x}_1 + \hat{x}_2 - \hat{x}_3) \\ \hline & 9\,\hat{y}_1 + 3\,\hat{y}_2 & \geq & (3\,\hat{y}_1 + 4\,\hat{y}_2)\,\hat{x}_1 + (2\,\hat{y}_1 + \hat{y}_2)\,\hat{x}_2 + (\hat{y}_1 - \hat{y}_2)\,\hat{x}_3. \end{array}$$

On the other hand, since $(\hat{y}_1, \hat{y}_2)$ is a feasible solution to the dual problem, we have $3\,\hat{y}_1 + 4\,\hat{y}_2 \geq 5$, $2\,\hat{y}_1 + \hat{y}_2 \geq 3$ and $\hat{y}_1 - \hat{y}_2 \geq -1$. Also, $\hat{x}_1 \geq 0$, $\hat{x}_2 \geq 0$ and $\hat{x}_3 \geq 0$, since $(\hat{x}_1, \hat{x}_2, \hat{x}_3)$ is a feasible solution to the primal problem. In this case, we obtain

$$\begin{array}{rrcl} & (3\,\hat{y}_1 + 4\,\hat{y}_2)\,\hat{x}_1 & \geq & 5\,\hat{x}_1 \\ & (2\,\hat{y}_1 + \hat{y}_2)\,\hat{x}_2 & \geq & 3\,\hat{x}_2 \\ + & (\hat{y}_1 - \hat{y}_2)\,\hat{x}_3 & \geq & -\hat{x}_3 \\ \hline & (3\,\hat{y}_1 + 4\,\hat{y}_2)\,\hat{x}_1 + (2\,\hat{y}_1 + \hat{y}_2)\,\hat{x}_2 + (\hat{y}_1 - \hat{y}_2)\,\hat{x}_3 & \geq & 5\,\hat{x}_1 + 3\,\hat{x}_2 - \hat{x}_3. \end{array}$$

Combining the two displayed inequalities we get

$$9\,\hat{y}_1 + 3\,\hat{y}_2 \geq (3\,\hat{y}_1 + 4\,\hat{y}_2)\,\hat{x}_1 + (2\,\hat{y}_1 + \hat{y}_2)\,\hat{x}_2 + (\hat{y}_1 - \hat{y}_2)\,\hat{x}_3 \geq 5\,\hat{x}_1 + 3\,\hat{x}_2 - \hat{x}_3.$$

So, we got $9\,\hat{y}_1 + 3\,\hat{y}_2 \geq 5\,\hat{x}_1 + 3\,\hat{x}_2 - \hat{x}_3$, saying that the objective value of the dual problem at the feasible dual solution $(\hat{y}_1, \hat{y}_2)$ is at least as large as the objective value of the primal problem at the feasible primal solution $(\hat{x}_1, \hat{x}_2, \hat{x}_3)$, which is exactly what weak duality says!

The moral of this story is that the objective value of the dual problem at any feasible solution to the dual problem is at least as large as the objective value of the primal problem at any feasible solution to the primal problem. This result is called weak duality. As discussed in the next section, weak duality has an important implication that allows us to check whether a pair of feasible solutions to the primal and dual problems are optimal to their respective problems.

5 Implication of Weak Duality

Assume that $(\hat{x}_1, \hat{x}_2, \hat{x}_3)$ is a feasible solution to the primal problem, whereas $(\hat{y}_1, \hat{y}_2)$ is a feasible solution to the dual problem in the previous section and these solutions yield the same objective function values for their respective problems in the sense that $5\,\hat{x}_1 + 3\,\hat{x}_2 - \hat{x}_3 = 9\,\hat{y}_1 + 3\,\hat{y}_2$. In this case, amazingly, we can use weak duality to immediately conclude that the solution $(\hat{x}_1, \hat{x}_2, \hat{x}_3)$ is optimal to the primal problem and the solution $(\hat{y}_1, \hat{y}_2)$ is optimal to the dual problem.

To see this result, let (x_1^*, x_2^*, x_3^*) be the optimal solution to the primal problem. Since $(\hat{x}_1, \hat{x}_2, \hat{x}_3)$ is a feasible, but not necessarily an optimal, solution to the primal problem, the objective value provided by the solution $(\hat{x}_1, \hat{x}_2, \hat{x}_3)$ for the primal problem cannot exceed the objective value provided by the optimal solution (x_1^*, x_2^*, x_3^*). So we have

On the other hand, let (y_1^*, y_2^*) be the optimal solution to the dual problem. Note that we minimize the objective function in the dual problem. Thus, since $(\hat{y}_1, \hat{y}_2)$ is a feasible, but not necessarily an optimal, solution to the dual problem, the objective value provided by the solution $(\hat{y}_1, \hat{y}_2)$ for the dual problem cannot dip below the objective value provided by the optimal solution (y_1^*, y_2^*). So, we also have

Lastly, since (x_1^*, x_2^*, x_3^*) is optimal to the primal problem, it is also a feasible solution to the primal problem. By the same reasoning, (y_1^*, y_2^*) is a feasible solution to the dual problem. Thus, since (x_1^*, x_2^*, x_3^*) is a feasible solution to the primal problem and (y_1^*, y_2^*) is a feasible solution to the dual problem, by weak duality, (y_1^*, y_2^*) provides an objective value for the dual problem that is at least as large as the objective value provided by (x_1^*, x_2^*, x_3^*) for the primal problem. In other words, we have

Combining the three displayed inequalities above, we obtain

If the solutions $(\hat{x}_1, \hat{x}_2, \hat{x}_3)$ and $(\hat{y}_1, \hat{y}_2)$ provide the same objective function values for their respective problems in the sense that $5\,\hat{x}_1 + 3\,\hat{x}_2 - \hat{x}_3 = 9\,\hat{y}_1 + 3\,\hat{y}_2$, then the left and right side of the inequalities above are the same. So, all of the terms in the inequalities have to be equal to each other. Thus, we get $5\,\hat{x}_1 + 3\,\hat{x}_2 - \hat{x}_3 = 5\,x_1^* + 3\,x_2^* - x_3^* = 9\,y_1^* + 3\,y_2^* = 9\,\hat{y}_1 + 3\,\hat{y}_2$. Having $5\,\hat{x}_1 + 3\,\hat{x}_2 - \hat{x}_3 = 5\,x_1^* + 3\,x_2^* - x_3^*$ implies that the objective value provided by the solution $(\hat{x}_1, \hat{x}_2, \hat{x}_3)$ for the primal problem is the same as the objective value provided by the optimal solution. In other words, the solution $(\hat{x}_1, \hat{x}_2, \hat{x}_3)$ is optimal for the primal problem. Likewise, having $9\,y_1^* + 3\,y_2^* = 9\,\hat{y}_1 + 3\,\hat{y}_2$ implies that the objective value provided by the solution $(\hat{y}_1, \hat{y}_2)$ for the dual problem is the same as the objective value provided by the dual solution. That is, the solution $(\hat{y}_1, \hat{y}_2)$ is optimal for the dual problem. This result is exactly what we set out to show!

The moral of this story is that if we have a feasible solution to the primal problem and a feasible solution to the dual problem and these feasible solutions provide the same objective function values for their respective problems, then the feasible solution we have for the primal problem is optimal for the primal problem and the feasible solution we have for the dual problem is optimal for the dual problem.

Strong Duality and Complementary Slackness

In the previous chapter, we saw an important implication of weak duality. In particular, if we have a feasible solution to the primal problem and a feasible solution to the dual problem and these feasible solutions provide the same objective function values for their respective problems, then the feasible solution we have for the primal problem is optimal for the primal problem and the feasible solution we have for the dual problem is optimal for the dual problem. In this chapter, we show that we automatically obtain the optimal solution to the dual problem when we solve the primal problem by using the simplex method.

1 Optimal Dual Solution from the Simplex Method

A surprising result is that if we solve the primal problem by using the simplex method, then we can automatically obtain the optimal solution for the dual problem by using the last system of equations that the simplex method reaches. To see this result, consider a primal linear problem and its corresponding dual given by

$$\begin{aligned} \max \quad & 5\,x_1 + 3\,x_2 - x_3 \\ \text{st} \quad & 3\,x_1 + 2\,x_2 + x_3 \leq 9 \\ & 4\,x_1 + x_2 - x_3 \leq 3 \\ & x_1, x_2, x_3 \geq 0, \end{aligned} \qquad \begin{aligned} \min \quad & 9\,y_1 + 3\,y_2 \\ \text{st} \quad & 3\,y_1 + 4\,y_2 \geq 5 \\ & 2\,y_1 + y_2 \geq 3 \\ & y_1 - y_2 \geq -1 \\ & y_1, y_2 \geq 0. \end{aligned}$$

For reference, we also write the versions of these linear programs with slack variables. Using the slack variables w_1 and w_2 for the primal constraints and the slack variables z_1, z_2 and z_3 for the dual constraints, the linear programs above are equivalent to

$$\begin{aligned} \max \quad & 5\,x_1 + 3\,x_2 - x_3 \\ \text{st} \quad & 3\,x_1 + 2\,x_2 + x_3 + w_1 = 9 \\ & 4\,x_1 + x_2 - x_3 + w_2 = 3 \\ & x_1, x_2, x_3, w_1, w_2 \geq 0, \end{aligned} \qquad \begin{aligned} \min \quad & 9\,y_1 + 3\,y_2 \\ \text{st} \quad & 3\,y_1 + 4\,y_2 - z_1 = 5 \\ & 2\,y_1 + y_2 - z_2 = 3 \\ & y_1 - y_2 - z_3 = -1 \\ & y_1, y_2, z_1, z_2, z_3 \geq 0. \end{aligned}$$

Consider solving the primal problem by using the simplex method. We start with the system of equations

$$\begin{array}{rcrcrcrcrcr} 5\,x_1 & + & 3\,x_2 & - & x_3 & & & & & = & \zeta \\ \hline 3\,x_1 & + & 2\,x_2 & + & x_3 & + & w_1 & & & = & 9 \\ 4\,x_1 & + & x_2 & - & x_3 & & & + & w_2 & = & 3. \end{array}$$

We increase x_1 up to $\min\{9/3, 3/4\} = 3/4$. Thus, the entering variable is x_1 and the leaving variable is w_2. Doing the appropriate row operations, the next system of equations is

$$
\begin{array}{rcrcrcrcrcl}
 & & \frac{7}{4}\,x_2 & + & \frac{1}{4}\,x_3 & & & - & \frac{5}{4}\,w_2 & = & \zeta - \frac{15}{4} \\
\hline
 & & \frac{5}{4}\,x_2 & + & \frac{7}{4}\,x_3 & + & w_1 & - & \frac{3}{4}\,w_2 & = & \frac{27}{4} \\
x_1 & + & \frac{1}{4}\,x_2 & - & \frac{1}{4}\,x_3 & & & + & \frac{1}{4}\,w_2 & = & \frac{3}{4}.
\end{array}
$$

We increase x_2 up to $\min\{\frac{27}{4}/\frac{5}{4}, \frac{3}{4}/\frac{1}{4}\} = 3$. So, the entering variable is x_2 and the leaving variable is x_1. Appropriate row operations yield the system of equations

$$
\begin{array}{rcrcrcrcrcl}
-7\,x_1 & & & + & 2\,x_3 & & & - & 3\,w_2 & = & \zeta - 9 \\
\hline
-5\,x_1 & & & + & 3\,x_3 & + & w_1 & - & 2\,w_2 & = & 3 \\
4\,x_1 & + & x_2 & - & x_3 & & & + & w_2 & = & 3.
\end{array}
$$

We increase x_3 up to $3/3 = 1$. In this case, the entering variable is x_3 and the leaving variable is w_1. Carrying out the necessary row operations gives

$$
\begin{array}{rcrcrcrcrcl}
-\frac{11}{3}\,x_1 & & & & & - & \frac{2}{3}\,w_1 & - & \frac{5}{3}\,w_2 & = & \zeta - 11 \\
\hline
-\frac{5}{3}\,x_1 & & & + & x_3 & + & \frac{1}{3}\,w_1 & - & \frac{2}{3}\,w_2 & = & 1 \\
-\frac{7}{3}\,x_1 & + & x_2 & & & + & \frac{1}{3}\,w_1 & + & \frac{1}{3}\,w_2 & = & 4.
\end{array}
$$

We reached the optimal solution. The solution $(x_1^*, x_2^*, x_3^*, w_1^*, w_2^*) = (0, 4, 1, 0, 0)$ is optimal for the primal linear program yielding the optimal objective value of 11.

Consider a possible solution $(y_1^*, y_2^*, z_1^*, z_2^*, z_3^*)$ to the dual problem constructed as follows. The values of the dual variables y_1 and y_2 are set to the negative of the objective function row coefficients of w_1 and w_2 in the final system of equations that the simplex method obtains. The values of the dual slack variables z_1, z_2 and z_3 are set to the negative of the objective function row coefficients of x_1, x_2 and x_3. Therefore, the solution $(y_1^*, y_2^*, z_1^*, z_2^*, z_3^*)$ is given by

$$
y_1^* = \frac{2}{3}, \quad y_2^* = \frac{5}{3}, \quad z_1^* = \frac{11}{3}, \quad z_2^* = 0, \quad z_3^* = 0.
$$

Note that this solution satisfies

Thus, the solution (y_1^*, y_2^*) is feasible to the dual problem and it provides the objective value of $9\,y_1^* + 3\,y_2^* = 9 \times \frac{2}{3} + 3 \times \frac{5}{3} = 11$ for the dual problem.

The solution $(x_1^*, x_2^*, x_3^*) = (0, 4, 1)$ is optimal for the primal problem providing an objective value of 11 for the primal problem. Therefore, $(x_1^*, x_2^*, x_3^*) = (0, 4, 1)$ is a feasible solution to the primal problem proving an objective value of 11 for the primal problem. The solution $(y_1^*, y_2^*) = (2/3, 5/3)$ is a feasible solution to the dual problem providing an objective value of 11 for the dual problem. So, we have a feasible solution to the primal problem and a feasible solution to the dual problem such that these feasible solutions provide the same objective function values for their respective problems. In this case, weak duality implies that the feasible solution we have for the primal problem is optimal for the primal problem and the feasible solution we have for the dual problem is optimal for the dual problem. Thus, the solution $(x_1^*, x_2^*, x_3^*) = (0, 4, 1)$ is optimal for the primal problem and the solution $(y_1^*, y_2^*) = (2/3, 5/3)$ is optimal for the dual problem. The fact that $(x_1^*, x_2^*, x_3^*) = (0, 4, 1)$ is optimal for the primal problem is no news to us. We knew that this solution is optimal for the primal problem. However, we now know that the solution $(y_1^*, y_2^*) = (2/3, 5/3)$ is optimal for the dual problem! Amazingly, we were able to just look at the objective function row coefficients of the final system of equations in the simplex method and construct an optimal solution to the dual problem by using these objective function row coefficients.

Also, consider the version of the dual problem with slack variables. For the solution $(y_1^*, y_2^*, z_1^*, z_2^*, z_3^*) = (2/3, 5/3, 11/3, 0, 0)$ constructed above, we have

$$\begin{aligned}
3\,y_1^* + 4\,y_2^* - z_1^* &= 3 \times \frac{2}{3} + 4 \times \frac{5}{3} - \frac{11}{3} = \frac{26}{3} - \frac{11}{3} = 5 \\
2\,y_1^* + y_2^* - z_2^* &= 2 \times \frac{2}{3} + \frac{5}{3} - 0 = 3 \\
y_1^* - y_2^* - z_3^* &= \frac{2}{3} - \frac{5}{3} - 0 = -1.
\end{aligned}$$

Therefore, the solution $(y_1^*, y_2^*, z_1^*, z_2^*, z_3^*) = (2/3, 5/3, 11/3, 0, 0)$ is feasible to the version of the dual problem with slack variables. In other words, the values of the dual slack variables z_1, z_2 and z_3 obtained by using the negative of the objective function row coefficients of the decision variables x_1, x_2 and x_3 give us the correct values of the slack variables in the dual solution $(y_1^*, y_2^*, z_1^*, z_2^*, z_3^*)$.

It seems magical to be able to use the objective function row coefficients in the final iteration of the simplex method to obtain an optimal solution to the dual problem. In the next section, we understand why we are able to obtain an optimal solution to the dual problem from the last system of equations that the simplex method reaches.

2 Strong Duality

In the previous section, we used the simplex method to obtain the optimal solution $(x_1^*, x_2^*, x_3^*, w_1^*, w_2^*) = (0, 4, 1, 0, 0)$ to the primal problem. The corresponding optimal objective value for the primal problem was 11. We used the final system of equations obtained by the simplex method to construct a solution $(y_1^*, y_2^*, z_1^*, z_2^*, z_3^*)$ to the dual problem. In this solution, the values of the dual variables y_1 and y_2 are set to the negative of the objective

function row coefficients of the primal slack variables w_1 and w_2 in the final system of equations. The values of the dual slack variables z_1, z_2 and z_3 are set to the negative of the objective function row coefficients of the primal variables x_1, x_2 and x_3. The solution $(y_1^*, y_2^*, z_1^*, z_2^*, z_3^*)$ obtained in this fashion is given by

$$y_1^* = \frac{2}{3}, \quad y_2^* = \frac{5}{3}, \quad z_1^* = \frac{11}{3}, \quad z_2^* = 0, \quad z_3^* = 0.$$

We showed that this solution satisfies three properties.

- First, the solution (y_1^*, y_2^*) is feasible to the dual problem, satisfying all of the constraints in the dual problem.
- Second, the solution (y_1^*, y_2^*) provides an objective value of 11 for the dual problem, which is the objective value provided by the solution (x_1^*, x_2^*, x_3^*) for the primal problem.
- Third, the solution $(y_1^*, y_2^*, z_1^*, z_2^*, z_3^*)$ is feasible to the version of the dual problem with slack variables.

Using the first two properties, we were able to conclude that the solution $(y_1^*, y_2^*, z_1^*, z_2^*, z_3^*)$ constructed by using the objective function row coefficients in the final iteration of the simplex method is optimal to the dual problem. In this section, we understand why the three properties above hold. The simplex method started with the system of equations

$$\begin{array}{rcrcrcrcrcr}
5\,x_1 & + & 3\,x_2 & - & x_3 & & & & & = & \zeta \\
\hline
3\,x_1 & + & 2\,x_2 & + & x_3 & + & w_1 & & & = & 9 \\
4\,x_1 & + & x_2 & - & x_3 & & & + & w_2 & = & 3.
\end{array}$$

The last system of equations in the simplex method was

$$\begin{array}{rcrcrcrcrcr}
-\frac{11}{3}\,x_1 & & & & & - & \frac{2}{3}\,w_1 & - & \frac{5}{3}\,w_2 & = & \zeta - 11 \\
\hline
-\frac{5}{3}\,x_1 & & & + & x_3 & + & \frac{1}{3}\,w_1 & - & \frac{2}{3}\,w_2 & = & 1 \\
-\frac{7}{3}\,x_1 & + & x_2 & & & + & \frac{1}{3}\,w_1 & + & \frac{1}{3}\,w_2 & = & 4.
\end{array}$$

The last system of equations is obtained by carrying out row operations starting from the initial system of equations. Therefore, we have to be able to obtain the objective function row in the last system of equations by multiplying the equations in the initial system of equations with some constants and adding them up.

In the objective function row in the last system of equations, w_1 appears with a coefficient of $-2/3$. Thus, if we are to obtain the objective function row in the last system of equations by multiplying the equations in the initial system of equations with some constants and adding them up, then we must multiply the first constraint row in the initial system of equations by $-2/3$ because w_1 appears nowhere else in the initial system of equations and there is no other way of having a coefficient of $-2/3$ for w_1 in the final system of

equations. By the same reasoning, we must multiply the second constraint row in the initial system of equations by $-5/3$. This argument indicates that if we are to obtain the objective function row in the last system of equations by multiplying the equations in the initial system of equations with some constants and adding them up, then we must multiply the first constraint row in the initial system of equations by $-2/3$ and the second constraint row by $-5/3$ and add them to the objective function row in the initial system of equations. We can check whether this assertion is actually correct. In particular, we can simply go ahead and multiply the first constraint row by $-2/3$ and the second constraint row by $-5/3$ and add them to the objective function row in the initial system of equations to see whether we get the objective function row in the last system of equations. Doing this calculation, we indeed obtain

$$\begin{array}{rrrrrrrrrrrrr}
 & 5\,x_1 & + & 3\,x_2 & - & x_3 & & & & = & \zeta \\
 & -\frac{2}{3} \times (3\,x_1 & + & 2\,x_2 & + & x_3 & + & w_1 &) & = & -\frac{2}{3} \times 9 \\
+ & -\frac{5}{3} \times (4\,x_1 & + & x_2 & - & x_3 & & & + \; w_2) & = & -\frac{5}{3} \times 3 \\
\hline
 & -\frac{11}{3}\,x_1 & + & 0\,x_2 & + & 0\,x_3 & - & \frac{2}{3}\,w_1 & - \; \frac{5}{3}\,w_2 & = & \zeta - 11,
\end{array}$$

which is exactly the objective function row in the final system of equations. The calculation above shows that the objective function row coefficient of x_1 in the final system of equations is obtained by multiplying the coefficients of x_1 in the two constraints in the initial system of equations by $-2/3$ and $-5/3$ and adding them to the objective function coefficient of x_1. We obtain the objective function row coefficients of x_2 and x_3 in the final system of equations by using a similar computation. Also, inspecting the calculation above, the objective function value of 11 in the final system of equations is obtained by multiplying the right side of the two constraints by $2/3$ and $5/3$ and adding them up. Therefore, we have

$$\begin{aligned}
5 - \frac{2}{3} \times 3 - \frac{5}{3} \times 4 &= -\frac{11}{3} \\
3 - \frac{2}{3} \times 2 - \frac{5}{3} \times 1 &= 0 \\
-1 - \frac{2}{3} \times 1 - \frac{5}{3} \times (-1) &= 0 \\
\frac{2}{3} \times 9 + \frac{5}{3}\,3 &= 11.
\end{aligned}$$

If we let $y_1^* = 2/3$, $y_2^* = 5/3$, $z_1^* = 11/3$, $z_2^* = 0$ and $z_3^* = 0$, then we can write the equalities above as

$$\begin{aligned}
5 - 3\,y_1^* - 4\,y_2^* &= -z_1^* \\
3 - 2\,y_1^* - y_2^* &= -z_2^* \\
-1 - y_1^* + y_2^* &= -z_3^* \\
9\,y_1^* + 3\,y_2^* &= 11.
\end{aligned}$$

Thus, rearranging the terms in the equalities above, if we let y_1^* and y_2^* be the negative of the objective function row coefficients of primal slack variables w_1 and w_2 in the final system

of equations and z_1^*, z_2^* and z_3^* be the negative of the objective function row coefficients of primal variables x_1, x_2 and x_3, then $(y_1^*, y_2^*, z_1^*, z_2^*, z_3^*)$ satisfies

$$\begin{aligned} 3\,y_1^* + 4\,y_2^* - z_1^* &= 5 \\ 2\,y_1^* + y_2^* - z_2^* &= 3 \\ y_1^* - y_2^* - z_3^* &= -1 \\ 9\,y_1^* + 3\,y_2^* &= 11. \end{aligned}$$

The first three equalities above show that $(y_1^*, y_2^*, z_1^*, z_2^*, z_3^*)$ satisfies the constraints in the version of the dual problem with slack variables. In addition, we note that the objective function row coefficients are all non-positive in the final iteration of the simplex method. Since the values of the decision variables $(y_1^*, y_2^*, z_1^*, z_2^*, z_3^*)$ are set to the negative of the objective function row coefficients, they are all non-negative. Thus, the solution $(y_1^*, y_2^*, z_1^*, z_2^*, z_3^*)$ is feasible to the version of the dual problem with slack variables, which establishes the third property that we set out to prove at the beginning of this section. On the other hand, the last equality above shows that the solution (y_1^*, y_2^*) provides the objective value of 11 for the dual problem, which is the objective value provided by the solution (x_1^*, x_2^*, x_3^*) for the primal problem, showing the second property at the beginning of this section. Finally, since $z_1^* \geq 0$, $z_2^* \geq 0$ and $z_3^* \geq 0$, the first three equalities above yield

$$\begin{aligned} 3\,y_1^* + 4\,y_2^* &= 5 + z_1^* \geq 5 \\ 2\,y_1^* + y_2^* &= 3 + z_2^* \geq 3 \\ y_1^* - y_2^* &= -1 + z_3^* \geq -1. \end{aligned}$$

Thus, the solution (y_1^*, y_2^*) is feasible to the dual problem, which establishes the first property at the beginning of this section.

As long as the simplex method terminates with an optimal solution, the objective function row coefficients in the final iteration will be all non-positive. In this case, we can always use the trick described in this chapter to construct an optimal solution to the dual problem by using the negative objective function row coefficients in the final iteration of the simplex method. The objective value provided by the solution that we obtain for the dual problem is always the same as the objective value provided by the solution that we have for the primal problem. We note that these observations will not hold when the simplex method does not terminate with an optimal solution, which is the case when there is no feasible solution to the problem or the problem is unbounded.

The moral of this story is the following. Consider a linear program with n decision variables and m constraints. Assume that the simplex method terminates with an optimal solution $(x_1^*, \ldots, x_n^*, w_1^*, \ldots, w_m^*)$ providing the optimal objective value of ζ^* for the primal problem. We construct a solution $(y_1^*, \ldots, y_m^*, z_1^*, \ldots, z_n^*)$ to the dual problem by letting y_i^* be the negative of the objective function row coefficient of w_i^* in the final iteration of the simplex method and z_j^* be the negative of the objective function row coefficient of x_j^* in the

final iteration of the simplex method. In this case, the solution $(y_1^*, \ldots, y_m^*, z_1^*, \ldots, z_n^*)$ is optimal for the dual problem. Furthermore, the solution $(y_1^*, \ldots, y_m^*, z_1^*, \ldots, z_n^*)$ provides an objective value of ζ^* for the dual problem. Thus, the optimal objective values of the primal and dual problems are equal. The last property is called strong duality.

In the previous chapter, we set out to construct to the dual problem to obtain an upper bound on the optimal objective value of the linear program we want to solve. Indeed, weak duality says that the objective value provided by any feasible solution to the dual problem is greater than or equal to the objective value provided by any feasible solution to the primal problem. This relationship holds for any pair of feasible solutions to the primal and dual problems. Strong duality, on the other hand, says that the objective value provided by the optimal solution to the dual problem is the same as the objective value provided by the optimal solution to the primal problem. Of course, strong duality holds when the primal problem has an optimal solution. That is, the primal problem is not infeasible or unbounded. Thus, strong duality says that as long as the primal problem is not infeasible or unbounded, the primal and dual problems have the same optimal objective values.

3 Complementary Slackness

Consider the primal and dual problem pair written with the slack variables (w_1, w_2) for the primal problem and (z_1, z_2, z_3) for the dual problem,

$$\begin{aligned} \max \quad & 5\,x_1 + 3\,x_2 - x_3 \\ \text{st} \quad & 3\,x_1 + 2\,x_2 + x_3 + w_1 = 9 \\ & 4\,x_1 + x_2 - x_3 + w_2 = 3 \\ & x_1, x_2, x_3, w_1, w_2 \geq 0, \end{aligned} \qquad \begin{aligned} \min \quad & 9\,y_1 + 3\,y_2 \\ \text{st} \quad & 3\,y_1 + 4\,y_2 - z_1 = 5 \\ & 2\,y_1 + y_2 - z_2 = 3 \\ & y_1 - y_2 - z_3 = -1 \\ & y_1, y_2, z_1, z_2, z_3 \geq 0. \end{aligned}$$

Assume that we have a solution $(\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{w}_1, \hat{w}_2)$ to the primal problem and a solution $(\hat{y}_1, \hat{y}_2, \hat{z}_1, \hat{z}_2, \hat{z}_3)$ to the dual problem satisfying the following three properties.

- The solution $(\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{w}_1, \hat{w}_2)$ is feasible for the primal problem and the solution $(\hat{y}_1, \hat{y}_2, \hat{z}_1, \hat{z}_2, \hat{z}_3)$ is feasible for the dual problem.
- We have $\hat{x}_j \times \hat{z}_j = 0$ for all $j = 1, 2, 3$.
- We have $\hat{y}_i \times \hat{w}_i = 0$ for all $i = 1, 2$.

It turns out satisfying the three properties above ensures that the solution $(\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{w}_1, \hat{w}_2)$ is optimal for the primal problem and the solution $(\hat{y}_1, \hat{y}_2, \hat{z}_1, \hat{z}_2, \hat{z}_3)$ is optimal for the dual problem. To see this result, it is enough to show that the objective value provided by the solution $(\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{w}_1, \hat{w}_2)$ for the primal problem is equal to the objective value

provided by the solution $(\hat{y}_1, \hat{y}_2, \hat{z}_1, \hat{z}_2, \hat{z}_3)$ for the dual problem. In this case, we have the solutions $(\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{w}_1, \hat{w}_2)$ and $(\hat{y}_1, \hat{y}_2, \hat{z}_1, \hat{z}_2, \hat{z}_3)$ such that these solutions are feasible for the primal and dual problems and they provide the same objective value for their respective problems. Therefore, by the implication of weak duality discussed at the end of the previous chapter, it must be the case that $(\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{w}_1, \hat{w}_2)$ is optimal for the primal problem and the solution $(\hat{y}_1, \hat{y}_2, \hat{z}_1, \hat{z}_2, \hat{z}_3)$ is optimal for the dual problem.

We proceed to showing that if the solutions $(\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{w}_1, \hat{w}_2)$ and $(\hat{y}_1, \hat{y}_2, \hat{z}_1, \hat{z}_2, \hat{z}_3)$ satisfy the three properties above, then they provide the same objective value for their respective problems. We have the chain of equalities

The first equality above uses the fact that $(\hat{y}_1, \hat{y}_2, \hat{z}_1, \hat{z}_2, \hat{z}_3)$ is feasible to the dual problem because of the first property above, which is to say that $3\,\hat{y}_1 + 4\,\hat{y}_2 - \hat{z}_1 = 5$, $2\,\hat{y}_1 + \hat{y}_2 - \hat{z}_2 = 3$ and $\hat{y}_1 - \hat{y}_2 - \hat{z}_3 = -1$. The second equality follows by arranging the terms. The third equality uses the fact that $\hat{x}_1\,\hat{z}_1 = \hat{x}_2\,\hat{z}_2 = \hat{x}_3\,\hat{z}_3 = \hat{y}_1\,\hat{w}_1 = \hat{y}_2\,\hat{w}_2 = 0$ by the second and third properties above. The fourth equality can be obtained by rearranging the terms. The last equality uses the fact that $(\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{w}_1, \hat{w}_2)$ is feasible to the primal problem by the first property above, which is to say that $3\,\hat{x}_1 + 2\,\hat{x}_2 + \hat{x}_3 + \hat{w}_1 = 9$ and $4\,\hat{x}_1 + \hat{x}_2 - \hat{x}_3 + \hat{w}_2 = 3$. So, the chain of equalities shows that the solutions $(\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{w}_1, \hat{w}_2)$ and $(\hat{y}_1, \hat{y}_2, \hat{z}_1, \hat{z}_2, \hat{z}_3)$ provide the same objective value for their respective problems. By the discussion in the previous paragraph, it must be the case that the solution $(\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{w}_1, \hat{w}_2)$ is optimal for the primal problem and the solution $(\hat{y}_1, \hat{y}_2, \hat{z}_1, \hat{z}_2, \hat{z}_3)$ is optimal for the dual problem.

The moral of this story is the following. Consider a linear program with n decision variables and m constraints. Assume that we have a feasible solution $(\hat{x}_1, \ldots, \hat{x}_n, \hat{w}_1, \ldots, \hat{w}_m)$ to the primal problem and a feasible solution $(\hat{y}_1, \ldots, \hat{y}_m, \hat{z}_1, \ldots, \hat{z}_n)$ to the dual problem. If these solutions satisfy

$$\begin{aligned} \hat{x}_j \times \hat{z}_j &= 0 \qquad \forall\, j = 1, \ldots, n \\ \hat{y}_i \times \hat{w}_i &= 0 \qquad \forall\, i = 1, \ldots, m, \end{aligned}$$

then the solution $(\hat{x}_1, \ldots, \hat{x}_n, \hat{w}_1, \ldots, \hat{w}_m)$ must be optimal for the primal problem and the solution $(\hat{y}_1, \ldots, \hat{y}_m, \hat{z}_1, \ldots, \hat{z}_n)$ must be optimal for the dual problem. This result is known as complementary slackness. Note that the first equality above can be interpreted as whenever $\hat{x}_j$ takes a strictly positive value, $\hat{z}_j$ is 0, and whenever $\hat{z}_j$ takes a strictly positive value, $\hat{x}_j$ is 0. A similar interpretation holds for the second equality above.

Why is complementary slackness useful? We can use complementary slackness to construct an optimal solution to the dual problem by using an optimal solution to the primal problem. In particular, assume that we solve the primal problem and see that $(x_1^*, x_2^*, x_3^*, w_1^*, w_2^*) = (0, 1, 4, 0, 0)$ is the optimal solution. Furthermore, x_2 and x_4 are the basic variables in the optimal solution. By using this information, we want to construct the optimal dual solution $(y_1^*, y_2^*, z_1^*, z_2^*, z_3^*)$ to the problem. To construct the optimal dual solution, by complementary slackness, it is enough to find $(y_1^*, y_2^*, z_1^*, z_2^*, z_3^*)$ such that $(y_1^*, y_2^*, z_1^*, z_2^*, z_3^*)$ is feasible to the dual problem and we have

$$x_1^* z_1^* = 0, \quad x_2^* z_2^* = 0, \quad x_3^* z_3^* = 0, \quad y_1^* w_1^* = 0, \quad y_2^* w_2^* = 0.$$

Since $x_1^* = w_1^* = w_2^* = 0$, we immediately have $x_1^* z_1^* = 0$, $y_1^* w_1^* = 0$ and $y_2^* w_2^* = 0$, irrespective of the values of z_1^*, y_1^* and y_2^*. Since $x_2^* = 1$ and $x_3^* = 4$, to satisfy $x_2^* z_2^* = 0$ and $x_3^* z_3^* = 0$, we must have $z_2^* = 0$ and $z_3^* = 0$. Also, since we want $(y_1^*, y_2^*, z_1^*, z_2^*, z_3^*)$ to be a feasible solution to the dual problem, we must have

Since we must have $z_2^* = 0$ and $z_3^* = 0$, the system of equations above is equivalent to

The system of equations above has three unknowns and three equations. Solving this system of equations, we obtain $y_1^* = 2/3$, $y_2^* = 5/3$ and $z_1^* = 11/3$. Thus, the solution $(x_1^*, x_2^*, x_3^*, w_1^*, w_2^*) = (0, 1, 4, 0, 0)$ is feasible for the primal problem, the solution $(y_1^*, y_2^*, z_1^*, z_2^*, z_3^*) = (\frac{2}{3}, \frac{5}{3}, \frac{11}{3}, 0, 0)$ is feasible for the dual problem and these solutions satisfy $x_1^* z_1^* = x_2^* z_2^* = x_3^* z_3^* = y_1^* w_1^* = y_2^* w_2^* = 0$. In this case, by complementary slackness, it follows that the solution $(x_1^*, x_2^*, x_3^*, w_1^*, w_2^*) = (0, 1, 4, 0, 0)$ is optimal for the primal problem and the solution $(y_1^*, y_2^*, z_1^*, z_2^*, z_3^*) = (\frac{2}{3}, \frac{5}{3}, \frac{11}{3}, 0, 0)$ is optimal for the dual problem.

Although we will not prove explicitly, it is possible to show that the converse of the statement in complementary slackness also holds. In particular, assume that we have a feasible solution $(x_1^*, \ldots, x_n^*, w_1^*, \ldots, w_m^*)$ to the primal problem and a feasible solution $(y_1^*, \ldots, y_m^*, z_1^*, \ldots, z_n^*)$ to the dual problem. If these solutions are optimal for their respective problems, then it must be the case that $\hat{x}_j \times \hat{z}_j = 0$ for all $j = 1, \ldots, n$ and $\hat{y}_i \times \hat{w}_i = 0$ for all $i = 1, \ldots, m$.

Economic Interpretation of the Dual Problem

One use of the dual problem is in finding an upper bound on the optimal objective value of a linear program we want to solve. In a previous chapter, we discussed how such an upper bound becomes useful when we try to understand the optimality gap of a feasible solution we have on hand before the simplex method reaches the optimal solution. In this chapter, we discuss another use of the dual problem. In particular, we see that the optimal solution to the dual problem can be used to understand how much the optimal objective value of the primal problem changes when we perturb the right sides of the constraints in the primal problem by small amounts. This information can be used to assess how valuable different resources are.

1 Motivation for Economic Analysis

Consider the following example. We sell cloud computing services to two classes of customers, memory-intensive and storage-intensive. Both customer classes are served through yearly contracts. Each memory-intensive customer takes up 100 GB of RAM and 200 GB of disk space, whereas each storage-intensive customer takes up 40 GB of RAM and 400 GB of disk space. From each yearly contract with a memory-intensive customer, we make \$2400. From each yearly contract with a storage-intensive customer, we make \$3200. We have 10000 GB of RAM and 60000 GB of disk space available to sign contracts with the two customer classes. For technical reasons, we do not want to get in a contract with more than 140 storage-intensive customers. We want to figure out how many yearly contracts to sign with customers from each class to maximize the yearly revenue. We can formulate this problem as a linear program. We use the decision variables x_1 and x_2 to respectively denote the number of contracts we sign with memory-intensive and storage-intensive customers. The problem we want to solve can be formulated as the linear program

$$\begin{array}{rcll} z^0 & = & \max & 2400\,x_1 + 3200\,x_2 \\ & & \text{st} & 100\,x_1 + 40\,x_2 \leq 10000 \\ & & & 200\,x_1 + 400\,x_2 \leq 60000 \\ & & & x_2 \leq 140 \\ & & & x_1, x_2 \geq 0. \end{array}$$

We use z^0 to denote the optimal objective value of the problem above, which corresponds to the optimal revenue we can obtain from yearly contracts in the current situation.

Assume that we can purchase additional disk space to enlarge our cloud computing business. We have a supplier that offers us to sell additional disk space at a cost of \$5 per GB for each year of use. Should we be willing to consider this offer? To answer this question, we assume that we have $60000 + \epsilon$ GB of disk space rather than 60000, where ϵ is a small amount. When we have $60000 + \epsilon$ GB of disk space, we can compute optimal revenue from

yearly contracts by solving the linear program

$$\begin{array}{rcll} z^\epsilon & = & \max & 2400\,x_1 + 3200\,x_2 \\ & & \text{st} & 100\,x_1 + 40\,x_2 \leq 10000 \\ & & & 200\,x_1 + 400\,x_2 \leq 60000 + \epsilon \\ & & & x_2 \leq 140 \\ & & & x_1, x_2 \geq 0. \end{array}$$

We use z^ϵ to denote the optimal objective value of the problem above, which corresponds to the optimal revenue we can obtain from yearly contracts when we have $60000 + \epsilon$ GB of disk space. If we have $z^\epsilon - z^0 \geq 5\,\epsilon$, then the increase in our yearly revenue with ϵ GB of extra disk space exceeds the cost of the ϵ GB of extra disk space we get from our supplier. Thus, we should be willing to consider the offer from our supplier, at least for a few GB of disk space. On the other hand, if we have $z^\epsilon - z^0 < 5\,\epsilon$, then the increase in our yearly revenue with ϵ GB of extra disk space is not worth the cost of the extra disk space. So, we should not consider the offer from our supplier. Note that $z^\epsilon - z^0$ corresponds to the change in the optimal objective value of the linear program when we increase the right side of the second constraint by a small amount ϵ.

The approach described above is a reasonable approach to assess the offer from our supplier, but it requires solving two linear programs, one to compute z^0 and one to compute z^ϵ. Perhaps solving two linear programs is not a big deal, but assume that an airline solves a linear program to assess the optimal revenue that it can obtain when it operates a certain set of flight legs with certain capacities on them. The airline wants to understand the revenue improvement from increasing the capacity on each one of its flight legs. If there are L flight legs in the network that the airline operates, then the airline may need to solve $1 + L$ linear programs, where the first linear program corresponds to the current situation and each one of the remaining L linear programs corresponds the case where we increase the capacity on each of the L flight legs by a small amount. If the airline network is large, then L can be large and solving these linear programs can consume a lot of time.

A natural question is whether we can get away with solving a single linear program to assess how much the optimal objective value of a linear program changes when we increase the right side of a constraint by a small amount. To answer this question, consider the linear program for the cloud computing example and its dual given by

$$\begin{array}{rll} \max & 2400\,x_1 + 3200\,x_2 & \\ \text{st} & 100\,x_1 + 40\,x_2 \leq 10000 & (y_1) \\ & 200\,x_1 + 400\,x_2 \leq 60000 & (y_2) \\ & x_2 \leq 140 & (y_3) \\ & x_1, x_2 \geq 0, & \end{array} \qquad \begin{array}{rl} \min & 10000\,y_1 + 60000\,y_2 + 140\,y_3 \\ \text{st} & 100\,y_1 + 200\,y_2 \geq 2400 \\ & 40\,y_1 + 400\,y_2 + y_3 \geq 3200 \\ & y_1, y_2, y_3 \geq 0. \end{array}$$

The dual variables y_1, y_2 and y_3 are respectively associated with the first, second and third

constraints in the primal problem. We are interested in understanding how much the optimal objective value of the primal problem above changes when we increase the right side of the second constraint by a small amount ϵ. We use (y_1^*, y_2^*, y_3^*) to denote the optimal solution to the dual problem. In the next section, we show that $y_2^* \epsilon$ is equal to the change in the optimal objective value of the primal problem above when we increase the right side of the second constraint by a small amount ϵ. Similarly, $y_1^* \epsilon$ and $y_3^* \epsilon$ respectively correspond to the change in the optimal objective value of the primal problem above when we increase the right side of the first and third constraints by a small amount ϵ.

Thus, we can solve the primal problem once by using the simplex method. From the previous chapter, we know how to obtain the optimal solution to the dual problem by using the final system of equations obtained by the simplex method. Letting (y_1^*, y_2^*, y_3^*) be the optimal solution to the dual problem, $y_1^* \epsilon$, $y_2^* \epsilon$ and $y_3^* \epsilon$ respectively correspond to the change in the optimal objective value of the primal problem when we increase the right side of the first, second and third constraints by a small amount ϵ. This discussion implies that we can solve the primal problem only once to figure out how much the optimal objective value of this problem would change when we increase the right side of any one of the constraints by a small amount!

2 Economic Analysis from the Dual Solution

Consider the primal and dual problems given by

$$\begin{array}{rll} \max & 2400\,x_1 + 3200\,x_2 & \\ \text{st} & 100\,x_1 + 40\,x_2 \le 10000 & (y_1) \\ & 200\,x_1 + 400\,x_2 \le 60000 & (y_2) \\ & x_2 \le 140 & (y_3) \\ & x_1, x_2 \ge 0, & \end{array} \qquad \begin{array}{rl} \min & 10000\,y_1 + 60000\,y_2 + 140\,y_3 \\ \text{st} & 100\,y_1 + 200\,y_2 \ge 2400 \\ & 40\,y_1 + 400\,y_2 + y_3 \ge 3200 \\ & y_1, y_2, y_3 \ge 0. \end{array}$$

Let (y_1^*, y_2^*, y_3^*) be the optimal solution to the dual problem. Our goal is to understand why $y_2^* \epsilon$ corresponds to the change in the optimal objective value of the primal problem when we increase the right side of the second constraint in the primal problem by a small amount ϵ. Consider solving the primal problem by using the simplex method. Using the slack variables w_1, w_2 and w_3 for the constraints in the primal problem, we start with the system of equations

$$\begin{array}{rcrcrcrcrcr} 2400\,x_1 & + & 3200\,x_2 & & & & & & & = & z \\ \hline 100\,x_1 & + & 40\,x_2 & + & w_1 & & & & & = & 10000 \\ 200\,x_1 & + & 400\,x_2 & + & & + & w_2 & & & = & 60000 \\ & & x_2 & + & & & & + & w_3 & = & 140. \end{array}$$

We increase x_2 up to $\min\{10000/40, 60000/400, 140\} = 140$. Thus, the entering variable is x_2 and the leaving variable is w_3. Doing the appropriate row operations, the next system of

equations is

$$\begin{array}{rrrrrrrrrrr}
2400\,x_1 & & & & & & & - & 3200\,w_3 & = & z - 448000 \\
\hline
100\,x_1 & & + & w_1 & & & & - & 40\,w_3 & = & 4400 \\
200\,x_1 & & & & + & w_2 & & - & 400\,w_3 & = & 4000 \\
& x_2 & & & & & & + & w_3 & = & 140.
\end{array}$$

We increase x_1 up to $\min\{4400/100, 4000/200\} = 20$. So, the entering variable is x_1 and the leaving variable is w_2. The necessary row operations yield the system of equations

$$\begin{array}{rrrrrrrrrr}
& & & - & 12\,w_2 & + & 1600\,w_3 & = & z - 496000 \\
\hline
& & w_1 & - & \tfrac{1}{2}\,w_2 & + & 160\,w_3 & = & 2400 \\
x_1 & & & + & \tfrac{1}{200}\,w_2 & - & 2\,w_3 & = & 20 \\
& x_2 & & & & + & w_3 & = & 140.
\end{array}$$

We increase w_3 up to $\min\{2400/160, 140\} = 15$. In this case, the entering variable is w_3 and the leaving variable is w_1. Appropriate row operations give the system of equations

$$\begin{array}{rrrrrrrrrr}
& & - & 10\,w_1 & - & 7\,w_2 & & & = & z - 520000 \\
\hline
& & & \tfrac{1}{160}\,w_1 & - & \tfrac{1}{320}\,w_2 & + & w_3 & = & 15 \\
x_1 & & + & \tfrac{1}{80}\,w_1 & - & \tfrac{1}{800}\,w_2 & & & = & 50 \\
& x_2 & - & \tfrac{1}{160}\,w_1 & + & \tfrac{1}{320}\,w_2 & & & = & 125.
\end{array}$$

All coefficients in the objective function row are non-positive. Thus, we obtained the optimal solution, which is given by $(x_1^*, x_2^*) = (50, 125)$. The optimal objective value is 520000. Furthermore, we know that if we define the solution (y_1^*, y_2^*, y_3^*) such that y_1^*, y_2^* and y_3^* are respectively the negative of the objective function row coefficients of w_1, w_2 and w_3 in the final system of equations, then the solution (y_1^*, y_2^*, y_3^*) is optimal to the dual. Therefore, the solution (y_1^*, y_2^*, y_3^*) with

$$y_1^* = 10, \qquad y_2^* = 7, \qquad y_3^* = 0$$

is optimal to the dual problem. We want to understand why $y_2^*\,\epsilon = 7\,\epsilon$ gives how much the optimal objective value of the primal problem changes when we increase the right side of the second constraint by a small amount ϵ.

Let us reflect on the row operations in the execution of the simplex method above. We started with the system of equations

$$\begin{aligned}
&2400\,x_1 + 3200\,x_2 = z \\
&100\,x_1 + 40\,x_2 + w_1 = 10000 \\
&200\,x_1 + 400\,x_2 + w_2 = 60000 \\
&x_2 + w_3 = 140.
\end{aligned}$$

After applying a sequence of row operations in the simplex method, we obtained the system of equations

$$\begin{aligned}
-10\,w_1 - 7\,w_2 &= z - 520000 \\
\frac{1}{160}\,w_1 - \frac{1}{320}\,w_2 + w_3 &= 15 \\
x_1 + \frac{1}{80}\,w_1 - \frac{1}{800}\,w_2 &= 50 \\
x_2 - \frac{1}{160}\,w_1 + \frac{1}{320}\,w_2 &= 125.
\end{aligned}$$

Consider replacing all of the appearances of w_2 in the two systems of equations above with $w_2 - \epsilon$. Therefore, if we start with the system of equations

and apply the same sequence of row operations on this system, then we would obtain the system of equations

Moving all of the terms that involve an ϵ to the right side, it follows that if we start with the system of equations

$$\begin{aligned}
2400\,x_1 + 3200\,x_2 &= z \\
100\,x_1 + 40\,x_2 + w_1 &= 10000 \\
200\,x_1 + 400\,x_2 + w_2 &= 60000 + \epsilon \\
x_2 + w_3 &= 140
\end{aligned}$$

and apply the same sequence of row operations that we did in the simplex method, then we

would obtain the system of equations

$$
\begin{aligned}
-10\,w_1 - 7\,w_2 &= z - 520000 - 7\,\epsilon \\
\frac{1}{160}\,w_1 - \frac{1}{320}\,w_2 + w_3 &= 15 - \frac{1}{320}\,\epsilon \\
x_1 + \frac{1}{80}\,w_1 - \frac{1}{800}\,w_2 &= 50 - \frac{1}{800}\,\epsilon \\
x_2 - \frac{1}{160}\,w_1 + \frac{1}{320}\,w_2 &= 125 + \frac{1}{320}\,\epsilon.
\end{aligned}
$$

Note that w_2 and ϵ have the same coefficient in each equation above. Now, consider solving the linear program after we increase the right side of the second constraint by a small amount ϵ. The simplex method starts with the system of equations

Starting from the system of equations above, let us apply the same sequence of row operations in the earlier execution of the simplex method. These row operations may or may not exactly be the ones followed by the simplex method when we solve the problem after we increase the right side of the second constraint by ϵ. Nevertheless, applying these row operations is harmless in the sense that we know that a system of equations remains unchanged when we apply a sequence of row operations on it. If we apply these row operations, then by the just preceding argument, we would obtain the system of equations

The objective function row coefficients are all non-positive in the system of equations above, which means that we reached the optimal solution. Thus, if we increase the right side of the second constraint by ϵ, then the optimal solution is given by $(x_1^*, x_2^*) = (50 - \frac{1}{800}\epsilon, 125 + \frac{1}{320}\,\epsilon)$ and the optimal objective value is $520000 + 7\,\epsilon$. As long as ϵ is small, we have $x_1^* = 50 - \frac{1}{800}\epsilon \geq 0$ and $x_2^* = 125 + \frac{1}{320}\,\epsilon \geq 0$. Also, for small ϵ, observe that

Thus, the solution $(x_1^*, x_2^*) = (50 - \frac{1}{800}\epsilon, 125 + \frac{1}{320}\,\epsilon)$ is feasible to the linear program when we increase the right side of the second constraint by a small amount ϵ.

If we increase the right side of the second constraint by a small amount ϵ, then the optimal objective value is $520000 + 7\,\epsilon$. If we do not change the right side of the second constraint at all, then the optimal objective value is 520000. Thus, $7\,\epsilon$ corresponds to the change in the optimal objective value when we increase the right side of the second constraint by a small amount ϵ. In the last system of equations the simplex method reaches, the coefficients of w_2 and ϵ in the objective function row are identical. Since y_2^* is given by the negative of the objective function row coefficient of w_2, we have $y_2^* = 7$, which is also the negative of the coefficient of ϵ in the objective function row. Thus, if we increase the right side of the second constraint by a small amount ϵ, then the change in the optimal objective value of the problem is given by $7\,\epsilon = y_2^*\,\epsilon$.

Roughly speaking, the moral of this story is the following. Consider a linear program with m constraints and let $(y_1^*, \ldots, y_m^*)$ be the optimal solution to the dual of this linear program. In this case, if we increase the right side of the i-th constraint by a small amount ϵ, then the optimal objective value of the linear program changes by $\epsilon\, y_i^*$.

3 An Exception to the Moral of the Story

There is an exception to the moral of this story. When we solved the original linear program by using the simplex method, the last system of equations was

$$\begin{array}{rrcrcrcrcr}
 & & - & 10\,w_1 & - & 7\,w_2 & & & = & z - 520000 \\
\hline
 & & & \frac{1}{160}\,w_1 & - & \frac{1}{320}\,w_2 & + & w_3 & = & 15 \\
x_1 & & + & \frac{1}{80}\,w_1 & - & \frac{1}{800}\,w_2 & & & = & 50 \\
 & x_2 & - & \frac{1}{160}\,w_1 & + & \frac{1}{320}\,w_2 & & & = & 125.
\end{array}$$

Also, when we increased the right side of the second constraint by a small amount ϵ and solved the linear program by using the simplex method, the last system of equations was

$$\begin{array}{rrcrcrcrcr}
 & & - & 10\,w_1 & - & 7\,w_2 & & & = & z - 520000 - 7\,\epsilon \\
\hline
 & & & \frac{1}{160}\,w_1 & - & \frac{1}{320}\,w_2 & + & w_3 & = & 15 - \frac{1}{320}\,\epsilon \\
x_1 & & + & \frac{1}{80}\,w_1 & - & \frac{1}{800}\,w_2 & & & = & 50 - \frac{1}{800}\,\epsilon \\
 & x_2 & - & \frac{1}{160}\,w_1 & + & \frac{1}{320}\,w_2 & & & = & 125 + \frac{1}{320}\,\epsilon.
\end{array}$$

The two systems of equations differ only in the right side of the equations. Now, assume that a basic variable, say x_1, took value 0 in the optimal solution in the first system of equations

above. In this case, the right side of the second constraint row in the first system of equations would be 0 instead of 50, whereas the right side of the second constraint row in the second system of equations would be $0 - \frac{1}{800}\,\epsilon = -\frac{1}{800}\epsilon$ instead of $50 - \frac{1}{800}\,\epsilon$. So, no matter how small ϵ is, setting $x_1 = -\frac{1}{800}\epsilon$ would yield a negative value for this decision variable and such a value for x_1 would be infeasible to the problem when we increase the right side of the second constraint by ϵ. Thus, we get into trouble when a basic variable at the optimal solution takes value 0. In other words, the moral of this story does not work when the optimal solution is degenerate.

Therefore, we need to refine the moral of this story as follows. Consider a linear program with n decision variables and m constraints. Let $(x_1^*, \ldots, x_n^*)$ be the optimal solution to the linear program and let $(y_1^*, \ldots, y_m^*)$ be the optimal solution to the dual of the linear program. Assume that $(x_1^*, \ldots, x_n^*)$ is not a degenerate solution. In this case, if we increase the right side of the i-th constraint by a small amount ϵ, then the optimal objective value of the linear program changes by $\epsilon\, y_i^*$.

Modeling Power of Integer Programming

Integer programs involve optimizing a linear objective function subject to linear constraints and with integrality requirements on the decision variables. Integer programs become useful when we deal with an optimization problem that includes indivisible quantities. For example, if we are building an optimization model to decide how many cars to ship from different production plants to different retailers, then we need to impose integrality constraints on our decision variables since a solution where we ship fractional numbers of cars would not be sensible. More importantly perhaps, integer programs become useful when we need to capture logical relationships between the decision variables. For example, we may be allowed to take an action only if we have taken another action earlier. Out of a certain number of actions available, we may be allowed to take only one of them. In this chapter, we use a number of examples to demonstrate how we can use integer programs to model optimization problems that fall outside the scope of linear programming.

1 Covering Problems

An ambulance organization of a city operates 2 ambulances. These ambulances can be stationed at any one of the 3 bases in the city. There are 5 districts in the city that needs coverage. The table below shows which bases provide coverage to which districts, where an entry of 1 corresponding to base i and district j indicates that an ambulance stationed at base i can cover district j. For example, if we station an ambulance at base 2, then we can cover districts 1, 3 and 5 with this ambulance. The populations of the 5 districts are respectively 1500, 3500, 2500, 3000 and 2000. We assume that covering a district with more than one ambulance does not bring any additional advantage over covering the district with a single ambulance. In other words, having one ambulance stationed at a based that covers a district is adequate to cover the population of the district. We want to decide where to station the ambulances so that we maximize the total population under coverage.

Base \ Dist.	1	2	3	4	5
1	1	1	0	1	0
2	1	0	1	0	1
3	0	1	0	0	1

To formulate the problem as an integer program, we use two sets of decision variables. The first set of decision variables captures whether we station an ambulance at each base. Thus, for all $i = 1, 2, 3$, we define the decision variable

$$x_i = \begin{cases} 1 & \text{if we station an ambulance at base } i \\ 0 & \text{otherwise.} \end{cases}$$

The second set of decision variables captures whether a district is under coverage. Therefore, for all $j = 1, 2, 3, 4, 5$, we define the decision variable

$$y_j = \begin{cases} 1 & \text{if we cover district } j \\ 0 & \text{otherwise.} \end{cases}$$

We have a logical relationship between the two types of decision variables. For example, district 5 can be covered only from bases 2 and 3, which implies that if we do not have an ambulance at bases 2 and 3, then district 5 is not covered. We can represent this relationship by the constraint $y_5 \leq x_2 + x_3$. Thus, if $x_2 = 0$ and $x_3 = 0$ so that we do not have an ambulance at bases 2 and 3, then it must be the case that $y_5 = 0$, which indicates that we cannot cover district 5. Using similar logical relationships for the coverage of other districts, we can figure out where to station the ambulances to maximize the total population under coverage by solving the integer program

In the objective function, we add up the populations of the districts that we cover. The first five constraints above ensure that if we do not have ambulances at any of the stations that cover a district, then we do not cover the district. For example, consider the constraint $y_5 \leq x_2 + x_3$. District 5 can be covered only from bases 2 and 3. If $x_2 = 0$ and $x_3 = 0$ so that there are no ambulances at bases 2 and 3, then the right side of the constraint $y_5 \leq x_2 + x_3$ is 0, which implies that we must have $y_5 = 0$. Thus, we do not cover district 5. On the other hand, if $x_2 = 1$ or $x_5 = 1$, then the right side of the constraint $y_5 \leq x_2 + x_3$ is at least 1, which implies that we can have $y_5 = 1$ or $y_5 = 0$. Since we are maximizing the objective function, the optimal solution would set $y_5 = 1$, which implies that we cover district 5. The last constraint in the problem above ensures that since we have 2 ambulances, we can station an ambulance at no more than 2 bases. We impose the requirement $x_i \in \{0, 1\}$ on the decision variable x_i for all $i = 1, 2, 3$. This requirement is equivalent to $0 \leq x_i \leq 1$ and x_i is an integer. The same argument holds for the decision variable y_j. Thus, the problem

above optimizes a linear objective function subject to linear constraints and with integrality requirements on the decision variables.

The integer program above is a specific instance of a covering problem. To give a compact formulation of covering problems, we consider the case where we have m possible actions and n goals. We can take at most k of the m possible actions. If we take action i, then we can satisfy a subset of the goals. To indicate which goals each action satisfies, we use

$$a_{ij} = \begin{cases} 1 & \text{if taking action } i \text{ satisfies goal } j \\ 0 & \text{otherwise.} \end{cases}$$

If we satisfy goal j, then we make a reward of R_j. Thus, the data for the problem is $\{a_{ij} : i = 1, \ldots, m,\ j = 1, \ldots, n\}$ and $\{R_j : j = 1, \ldots, n\}$. We want to figure out which actions to take to maximize the reward from the satisfied goals while making sure that we do not take more than k actions. To draw parallels with our previous example, action i corresponds to stationing an ambulance at base i and goal j corresponds to covering district j. In the data, a_{ij} indicates whether an ambulance at base i covers district j or not, whereas R_j corresponds to the population of district j. To formulate the problem as an integer program, we define the decision variables

$$x_i = \begin{cases} 1 & \text{if we take action } i \\ 0 & \text{otherwise,} \end{cases}$$

$$y_j = \begin{cases} 1 & \text{if we satisfy goal } j \\ 0 & \text{otherwise.} \end{cases}$$

In this case, the compact formulation of the covering problem is given by

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} R_j\, y_j \\ \text{st} \quad & y_j \le \sum_{i=1}^{m} a_{ij}\, x_i && \forall\, j = 1, \ldots, n \\ & \sum_{i=1}^{m} x_i \le k \\ & x_i \in \{0,1\},\ y_j \in \{0,1\} && \forall\, i = 1, \ldots, m,\ j = 1, \ldots, n. \end{aligned}$$

Note that a_{ij} takes value 1 if action i satisfies goal j, otherwise a_{ij} takes value 0. Thus, the sum $\sum_{i=1}^{m} a_{ij}\, x_i$ on the right side of the first constraint corresponds to the number of actions that we take satisfying goal j. If we do not take any actions that satisfy goal j so that $\sum_{i=1}^{m} a_{ij}\, x_i = 0$, then we must have $y_j = 0$, indicating that we cannot satisfy goal j. The second constraint ensures that we take at most k actions.

2 Fixed Charge Problems

We need to produce a product to satisfy a demand of 1000 units. There are 4 facilities that we can use to produce the product. If we use a certain facility to produce the product, then we pay a fixed charge for our usage of the facility, which does not depend on how many units we produce. Along with the fixed charge, we also incur a per unit production cost for each produced unit. The table below shows the fixed change and the per unit production cost when we produce the product at each one of the 4 facilities. For example, if we decide to produce the product in facility 1, then we incur a fixed charge of \$500 and for each unit that we produce at facility 1, we incur a per unit production cost of \$4. There is a production capacity of 500 at each facility, which is to say that we cannot produce more than 500 units at any one of the facilities. We want to figure out how many units to produce at each facility to minimize the total production cost, while making sure that we produce enough to satisfy the demand of 1000 units.

Facility	1	2	3	4
Fixed Charge	500	1200	800	500
Per Unit Cost	4	2	3	5

We formulate the problem by using a mixture of integer and continuous decision variables. For all $j = 1, 2, 3, 4$, we define the decision variable

$$x_j = \begin{cases} 1 & \text{if we use facility } j \text{ for production} \\ 0 & \text{otherwise.} \end{cases}$$

Also, for all $j = 1, 2, 3, 4$, we define the decision variable

y_j = Production quantity at facility j.

If $x_j = 0$, which means that we do not use facility j for production, then we must have $y_j = 0$ as well, indicating that the amount produced at facility j must be 0. On the other hand, if $x_j = 1$, meaning that we use facility j for production, then y_j is upper bounded by the capacity at the production facility, which is 500. Thus, we can represent the relationship between x_j and y_j by using the constraint $y_j \leq 500\, x_j$. In this case, we can figure out how many units to produce at each facility by solving the integer program

In the objective function above, we use the decision variable x_j to account for the fixed charge of using facility j and the decision variable y_j to account for the cost incurred for the units that we produce at facility j. The last constraint ensures that the total production quantity is enough to cover the demand.

The integer program above is a fixed charge problem, where we incur a fixed charge to produce a product at a particular facility. To give a compact formulation for a fixed charge problem, consider the case where we have n facilities. The fixed charge for using facility j is f_j and the per unit production cost at facility j is c_j. We use U_j to denote the production capacity of facility j. We need to produce enough to cover a demand of D units. We want to decide how much to produce at each facility to minimize the total fixed charges and production costs. Using the decision variables x_j and y_j as defined earlier in this section, the compact formulation of the fixed charge problem is given by

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} f_j\, x_j + \sum_{j=1}^{n} c_j\, y_j & \\ \text{st} \quad & y_j \le U_j\, x_j & \forall\, j = 1, \ldots, n \\ & \sum_{j=1}^{n} y_j \ge D & \\ & x_j \in \{0,1\},\ y_j \ge 0 & \forall\, j = 1, \ldots, n. \end{aligned}$$

If there is no production capacity at facility j, then we can replace the constraint $y_j \le U_j\, x_j$ with $y_j \le M\, x_j$ for some large number M. In the problem above, we know that we will never produce more than D units at a production facility. So, using $M = D$ suffices.

3 Problems with Either-Or Constraints

We can purchase a product from 4 different suppliers. The total amount we want to purchase from these 4 suppliers is 100. The price charged by each supplier is different. Furthermore, the distance from each supplier to our business is different and we want to make sure that the average distance that all of our purchases travel is no larger than 400 miles. Lastly, the suppliers are not willing to supply intermediate amounts of product. Our purchase quantities should be either rather small or rather large. In particular, the amount that we purchase from each supplier should either be smaller than a low threshold or larger than a high threshold. The table below shows the prices charged by the suppliers and their distances from our business, along with the low thresholds and high thresholds for the purchase quantities. For example, supplier 1 charges a price of \$5 for each unit of product, it is located 450 miles from our business and the purchase quantity from supplier 1 should be either less than 10 or larger than 50. We want to figure out how many units to purchase

from each supplier so that we minimize the total cost of the purchases, while making sure that we purchase a total of at least 100 units, the average distance traveled by all purchases is no larger than 400 miles and the purchase quantity from each supplier is either smaller than the low threshold or larger than the high threshold. It is important to observe that this problem requires modeling constraints that have either-or form, since the quantity that we purchase from a supplier has to be either smaller than the low threshold or larger than the high threshold.

Supplier	1	2	3	4
Price	5	6	3	7
Distance	450	700	800	200
Low Thresh.	10	15	5	10
High Thresh.	50	40	30	45

We formulate the problem by using a mixture of integer and continuous decision variables. For all $j = 1, 2, 3, 4$, we define the decision variable

$$x_j = \begin{cases} 1 & \text{if the purchase quantity from supplier } j \text{ is smaller than the low threshold} \\ 0 & \text{otherwise.} \end{cases}$$

Also, for all $j = 1, 2, 3, 4$, we define the decision variable

$$y_j = \text{Purchase quantity from supplier } j.$$

Consider the amount that we purchase from supplier 1. If $x_1 = 1$, then the purchase quantity from supplier 1 is smaller than the low threshold, which implies that we must have $y_1 \leq 10$. On the other hand, if $x_1 = 0$, then the purchase quantity from supplier 1 is larger the high threshold, which implies that we must have $y_1 \geq 50$. To capture this relationship between the decision variables x_1 and y_1, we use the two constraints

$$y_1 \leq 10 + M\,(1 - x_1) \quad \text{and} \quad y_1 \geq 50 - M x_1,$$

where M is a large number. In this case, if $x_1 = 1$, then the two constraints above take the form $y_1 \leq 10$ and $y_1 \geq 50 - M$. Since M is a large number, $50 - M$ is a small number. Thus, $y_1 \geq 50 - M$ is always satisfied. Thus, if $x_1 = 1$, then we must have $y_1 \leq 10$, as desired. On the other hand, if $x_1 = 0$, then the two constraints above take the form $y_1 \leq 10 + M$ and $y_1 \geq 50$. Since M is a large number, $y_1 \leq 10 + M$ is always satisfied. Thus, if $x_1 = 0$, then we must have $y_1 \geq 50$, as desired. We can follow a similar reasoning to ensure that the purchase quantities from the other suppliers are either smaller than the low threshold or larger than the high threshold.

Another constraint we need to impose on our decisions is that the average distance that all of our purchases travels is no larger than 400 miles. The average distance traveled by all of

our purchases is given by $(450\,y_1+700\,y_2+800\,y_3+200\,y_4)/(y_1+y_2+y_3+y_4)$. Thus, we want to ensure that $(450\,y_1+700\,y_2+800\,y_3+200\,y_4)/(y_1+y_2+y_3+y_4) \leq 400$. This constraint appears to be nonlinear since we have a fraction on the left side, but we can write this constraint equivalently as $450\,y_1+700\,y_2+800\,y_3+200\,y_4 \leq 400\,(y_1+y_2+y_3+y_4)$. Collecting all terms on one side of the inequality, we can ensure that the average distance that all of our purchases travels is no larger than 400 miles by using the constraint

$$50\,y_1+300\,y_2+400\,y_3-200\,y_4 \leq 0.$$

Putting the discussion so far together, we can figure out how may units to purchase from each supplier by solving the integer program

The objective function above accounts for the total cost of the purchases. The first eight constraints ensure that the purchase quantity from each supplier should be either smaller than the low threshold or larger than the high threshold. The last two constraints ensure that the average distance traveled by our orders is no larger than 400 miles and the total quantity we purchase is at least the desired amount of 100.

The integer program above involves either-or constraints. In particular, out of two constraints, we need to satisfy either one constraint or the other but not necessarily both. To describe a more general form of either-or constraints, consider a linear program with n non-negative decision variables $\{y_j : j = 1, \ldots, n\}$. In the objective function of

the linear program, we maximize $\sum_{j=1}^{n} c_j\, y_j$ for appropriate objective function coefficients $\{c_j : j = 1, \ldots, n\}$. For all $i = 1, \ldots, m$, we have the constraints

$$\sum_{j=1}^{n} a_{ij}\, x_j \leq b_i$$

for appropriate constraint coefficients $\{a_{ij} : i = 1, \ldots, m,\ j = 1, \ldots, n\}$ and constraint right sides $\{b_i : i = 1, \ldots, m\}$. Out of the m constraints above, we want at least k of them to be satisfied. Therefore, our goal is to maximize the objective function $\sum_{j=1}^{n} c_j\, y_j$ subject to the constraint that at least k out of the m constraints above are satisfied. To formulate this problem as an integer program, in addition to the decision variables $\{y_j : j = 1, \ldots, n\}$, we define the decision variables $\{x_i : i = 1, \ldots, m\}$ such that

$$x_i = \begin{cases} 1 & \text{if the constraint } \sum_{j=1}^{n} a_{ij}\, y_j \leq b_i \text{ is satisfied} \\ 0 & \text{otherwise.} \end{cases}$$

In our integer program, letting M be a larger number, we replace the constraint $\sum_{j=1}^{n} a_{ij}\, y_j \leq b_i$ with the constraint

$$\sum_{j=1}^{n} a_{ij}\, y_j \leq b_i + M\,(1 - x_i).$$

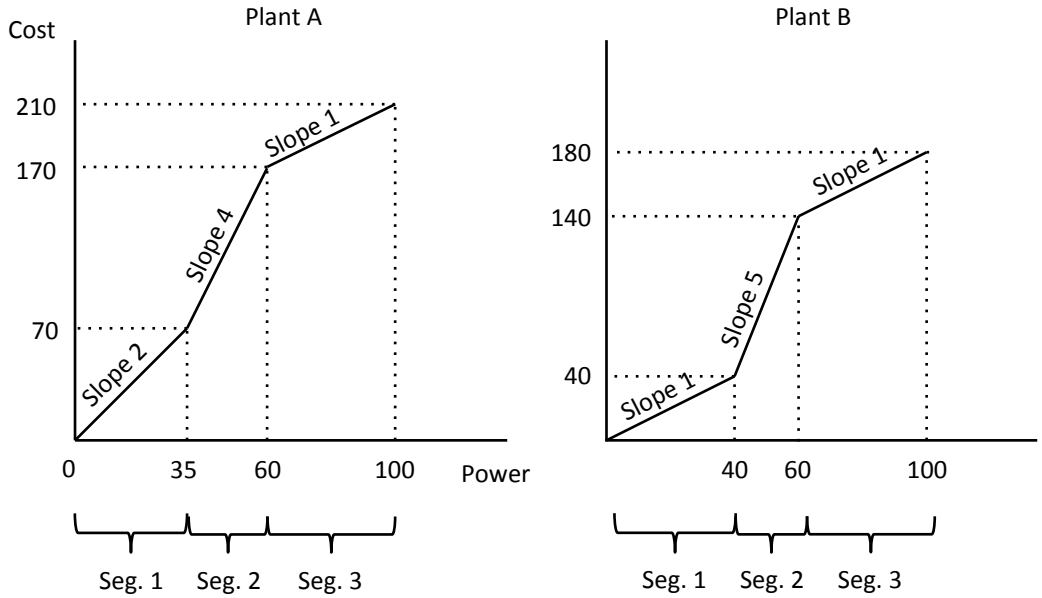
In this case, we can maximize the objective function $\sum_{j=1}^{n} c_j\, y_j$ subject to the constraint that at least k out of the m constraints are satisfied by solving the integer program

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} c_j\, y_j \\ \text{st} \quad & \sum_{j=1}^{n} a_{ij}\, y_j \leq b_i + M\,(1 - x_i) && \forall\, i = 1, \ldots, m \\ & \sum_{i=1}^{m} x_i \geq k \\ & x_i \in \{0, 1\},\ y_j \geq 0 && \forall\, i = 1, \ldots, m,\ j = 1, \ldots, n. \end{aligned}$$

In the problem above, consider the constraint $\sum_{j=1}^{n} a_{ij}\, y_j \leq b_i + M\,(1 - x_i)$. If $x_i = 1$, then the constraint takes the form $\sum_{j=1}^{n} a_{ij}\, y_j \leq b_i$. Thus, if $x_i = 1$, then we ensure that the decision variables $\{y_j : j = 1, \ldots, n\}$ satisfy the constraint $\sum_{j=1}^{n} a_{ij}\, y_j \leq b_i$. If $x_i = 0$, then the constraint takes the form $\sum_{j=1}^{n} a_{ij}\, y_j \leq b_i + M$, which is a constraint that is always satisfied since $b_i + M$ is a large number. Thus, if $x_i = 0$, we do not care whether the decision variables $\{y_j = 1, \ldots, n\}$ satisfy the constraint $\sum_{j=1}^{n} a_{ij}\, y_j \leq b_i$. The second constraint above imposes the condition that the decision variables $\{y_j = 1, \ldots, n\}$ must satisfy at least k of the constraints $\sum_{j=1}^{n} a_{ij}\, y_j \leq b_i$ for all $i = 1, \ldots, m$.

4 Problems with Nonlinear Objectives

We are generating power at 2 power plants, plants A and B. We want to generate a total of 100 units of power from the 2 plants. The cost of power generated at a power plant is a nonlinear function of the amount of power that we generate at the plant. The figure below shows the cost of power generated at each one of the 2 plants as a function of the power generated. For example, for plant A, for the first 35 units of power generated, each additional unit of power generation costs \$2. For the next 25 units of power generated, each additional unit of power generation costs \$4. Lastly, for the next 40 units of power generated, each additional unit of power generation costs \$1. We want to figure out how much power to generate at each plant to minimize the cost of generation, while making sure that we generate a total of 100 units of power. Note that the cost of power is nonlinear in the power generated. So, this problem involves minimizing a nonlinear function.



To understand the decision variables that we need, we focus on plant A and take a closer look at the graph in the figure above that gives the cost of generation as a function of the power generated. There are three segments in the horizontal axis of the graph and these three segments are labeled as 1, 2 and 3. For each one of the segments $i = 1, 2, 3$, we define the decision variable

$$x_{iA} = \begin{cases} 1 & \text{if the power generated at plant } A \text{ utilizes segment } i \\ 0 & \text{otherwise.} \end{cases}$$

For example, if we generate 45 units of power at plant A, then $x_{1A} = 1$, $x_{2A} = 1$ and $x_{3A} = 0$. Also, for each one of the segments $i = 1, 2, 3$, we define the decision variable

y_{iA} = Portion of segment i utilized by the power generated at plant A.

For example, if we generate 45 units of power at plant A, then the decision variables y_{1A}, y_{2A} and y_{3A} take the values $y_{1A} = 35$, $y_{2A} = 10$ and $y_{3A} = 0$.

Note that as a function of the decision variables y_{1A}, y_{2A} and y_{3A}, the total amount of power generated at plant A is given by

$$y_{1A} + y_{2A} + y_{3A}.$$

For each unit of power generated in segment 1, we incur an additional cost of \$2. For each unit of power generated in segment 2, we incur an additional cost of \$4. Finally, for each unit of power generated in segment 3, we incur an additional cost of \$1. Therefore, we can write the total cost of power generated at plant A as

$$2\,y_{1A} + 4\,y_{2A} + 1\,y_{3A}.$$

On the other hand, if $x_{1A} = 1$, then we use segment 1 when generating power at plant A. In this case, noting that the width of segment 1 is 35, we must have $y_{1A} \leq 35$. If $x_{1A} = 0$, then we do not use segment 1 when generating power at plant A. In this case, we must have $y_{1A} = 0$. To capture this relationship, we use the constraint $y_{1A} \leq 35\,x_{1A}$. Note that since $x_{1A} \in \{0, 1\}$, this constraint implies that we always have $y_{1A} \leq 35$. Furthermore, if $x_{2A} = 1$, then we use segment 2 when generating power at plant A, which means that we must have use segment 1 in its entirety. Therefore, if $x_{2A} = 1$, then we must have $y_{1A} \geq 35$. To capture this relationship, we use the constraint $y_{1A} \geq 35\,x_{2A}$. Thus, the decision variable y_{1A} is connected to the decision variables x_{1A} and x_{2A} through the constraints

$$y_{1A} \leq 35\,x_{1A} \quad \text{and} \quad y_{1A} \geq 35\,x_{2A}.$$

By using the same argument for segment 2, the decision variable y_{2A} is connected to the decision variables x_{2A} and x_{3A} through the constraints

$$y_{2A} \leq 25\,x_{2A} \quad \text{and} \quad y_{2A} \geq 25\,x_{3A}.$$

Lastly, if $x_{3A} = 1$, then we use segment 3 when generating power at plant A so that $y_{3A} \leq 40$. If $x_{3A} = 0$, then we do not use segment 3 when generating power at plant A so that we must have $y_{3A} = 0$. To capture this relationship, we use the constraint

$$y_{3A} \leq 40\,x_{3A}.$$

We can use the same approach to capture the cost of power generation at plant B. For each one of the segments $i = 1, 2, 3$, we define the decision variable

$$x_{iB} = \begin{cases} 1 & \text{if the power generated at plant } B \text{ utilizes segment } i \\ 0 & \text{otherwise.} \end{cases}$$

Also, for each one of the segments $i = 1, 2, 3$, we define the decision variable

y_{iB} = Portion of segment i utilized by the power generated at plant B.

In this case, the total amount of power generated at plant B is given by $y_{1B} + y_{2B} + y_{3B}$, whereas the total cost of power generated at plant B is given by $1\,y_{1B} + 5\,y_{2B} + 1\,y_{3B}$. By using the same approach that we used for plant A, the decision variables y_{1B}, y_{2B} and y_{3B} are connected to the decision variables x_{1B}, x_{2B} and x_{3B} through the constraints

$$y_{1B} \leq 40\,x_{1B}, \quad y_{1B} \geq 40\,x_{2B}, \quad y_{2B} \leq 20\,x_{2B}, \quad y_{2B} \geq 20\,x_{3B}, \quad y_{3B} \leq 40\,x_{3B}.$$

Collecting all of our discussion so far together, if we want to figure out how much power to generate at each plant to generate a total of 100 units of power with minimum generation cost, then we can solve the integer program

The integer program above provides an approach for dealing with single-dimensional piecewise-linear objective functions in our optimization problems. Any single-dimensional nonlinear function can be approximated arbitrarily well with a piecewise-linear function. Therefore, by using the approach described in this section, we can use rather accurate approximations of single-dimensional nonlinear functions as objective functions in our optimization problems.

Branch-and-Bound Method for Solving Integer Programs

In the previous chapter, we discussed a variety of optimization problems that can be modeled as integer programs. In this chapter, we discuss the branch-and-bound method for solving integer programs. The branch-and-bound method obtains the optimal solution to an integer program by solving a sequence of linear programs. Since the branch-and-bound method obtains the optimal solution to an integer program by solving a sequence of linear programs, it allows us to build on the theory and the algorithms that we already have for solving linear programs.

1 Key Idea of the Branch-and-Bound Method

Consider the integer program

$$\begin{aligned} \max \quad & 5\,x_1 + 4\,x_2 + 4\,x_3 + 3\,x_4 \\ \text{st} \quad & 2\,x_1 + 4\,x_2 + 3\,x_3 + 2\,x_4 \leq 20 \\ & 6\,x_1 + 5\,x_2 + 4\,x_3 + 5\,x_4 \leq 25 \\ & x_1 + x_2 + x_3 + x_4 \geq 5 \\ & x_2 + 2\,x_3 \leq 7 \\ & x_1, x_2, x_3, x_4 \geq 0 \\ & x_1, x_2, x_3 \text{ are integers.} \end{aligned}$$

Note that the decision variables x_1, x_2 and x_3 in the problem above are restricted to be integers but the decision variable x_4 can take fractional values. In the branch-and-bound method, we start by solving the integer program above without paying attention to any of the integrality requirements. In particular, we start by solving the problem

$$\begin{aligned} \max \quad & 5\,x_1 + 4\,x_2 + 4\,x_3 + 3\,x_4 \\ \text{st} \quad & 2\,x_1 + 4\,x_2 + 3\,x_3 + 2\,x_4 \leq 20 \\ & 6\,x_1 + 5\,x_2 + 4\,x_3 + 5\,x_4 \leq 25 \\ & x_1 + x_2 + x_3 + x_4 \geq 5 \\ & x_2 + 2\,x_3 \leq 7 \\ & x_1, x_2, x_3, x_4 \geq 0. \end{aligned}$$

The problem above is referred to as the linear programming relaxation of the integer program we want to solve. Since there are no integrality requirements on the decision variables, we can solve the problem above by using the simplex method. The optimal objective value of the problem above is 23.167 with the optimal solution

$$x_1 = 1.833,\ x_2 = 0,\ x_3 = 3.5,\ x_4 = 0.$$

The solution above satisfies the first four constraints in the integer program because these constraints are already included in the linear program that we just solved. However, the solution above is not a feasible solution to the integer program that we want to solve because the decision variables x_1 and x_3 take fractional values in the solution, whereas our integer program imposes integrality constraints on these decision variables. We focus on one of these decision variables, say x_1. We have $x_1 = 1.833$ in the solution above. Note that in the optimal solution to the integer program, we must have either $x_1 \leq 1$ or $x_1 \geq 2$. Thus, based on the optimal solution of the linear program that we just solved, we consider two cases. The first case focuses on $x_1 \leq 1$ and the second case focuses on $x_1 \geq 2$. These two cases yield two linear programs to consider, where the first linear program imposes the additional constraint $x_1 \leq 1$ and the second linear program imposes the additional constraint $x_1 \geq 2$. Thus, these two linear programs are given by

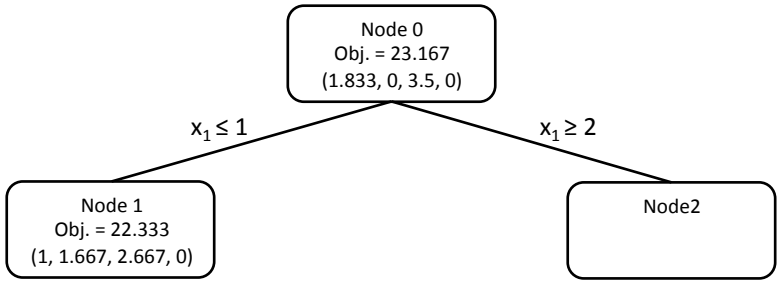
$$\begin{aligned} \max \quad & 5\,x_1 + 4\,x_2 + 4\,x_3 + 3\,x_4 \\ \text{st} \quad & 2\,x_1 + 4\,x_2 + 3\,x_3 + 2\,x_4 \leq 20 \\ & 6\,x_1 + 5\,x_2 + 4\,x_3 + 5\,x_4 \leq 25 \\ & x_1 + x_2 + x_3 + x_4 \geq 5 \\ & x_2 + 2\,x_3 \leq 7 \\ & x_1 \leq 1 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned} \qquad \begin{aligned} \max \quad & 5\,x_1 + 4\,x_2 + 4\,x_3 + 3\,x_4 \\ \text{st} \quad & 2\,x_1 + 4\,x_2 + 3\,x_3 + 2\,x_4 \leq 20 \\ & 6\,x_1 + 5\,x_2 + 4\,x_3 + 5\,x_4 \leq 25 \\ & x_1 + x_2 + x_3 + x_4 \geq 5 \\ & x_2 + 2\,x_3 \leq 7 \\ & x_1 \geq 2 \\ & x_1, x_2, x_3, x_4 \geq 0. \end{aligned}$$

An important observation is that the optimal solution to either of the two linear programs above will necessarily be different from the optimal solution to the linear program that we just solved because noting the constraints $x_1 \leq 1$ and $x_1 \geq 2$ in the two linear programs above, having $x_1 = 1.833$ in a solution would be infeasible to either of the two linear programs. Solving the linear program on the left above, the optimal objective value is 22.333 with the optimal solution

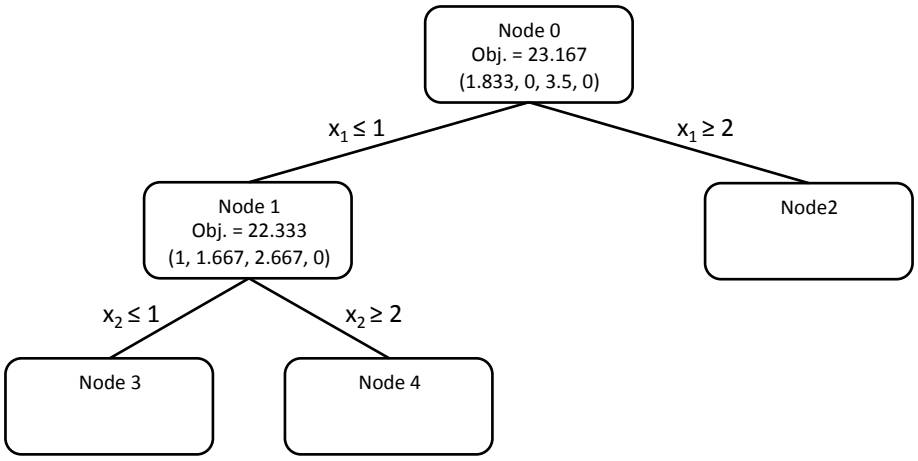
$$x_1 = 1,\ x_2 = 1.667,\ x_3 = 2.667,\ x_4 = 0.$$

We summarize our progress so far in the figure below. We started with the linear programming relaxation to the original integer program that we want to solve. This linear programming relaxation corresponds to node 0 in the figure. The optimal solution to the linear program at node 0 is $(x_1, x_2, x_3, x_4) = (1.833, 0, 3.5, 0)$ with the objective value 23.167. Observe how we display this solution and the objective value at node 0 in the figure below. Examining this solution, since the integer decision variable x_1 takes the fractional value 1.833 in the solution, we branch into two cases, $x_1 \leq 1$ and $x_1 \geq 2$. Branching into these two cases gives us the linear programs at nodes 1 and 2 in the figure. The linear program at node 1 includes all of the constraints in the linear program at node 0, along with the constraint $x_1 \leq 1$. The linear program at node 2 includes all of the constraints in the linear program at node 0, along with the constraint $x_1 \geq 2$. Solving the linear program at

node 1, we obtain the optimal solution $(x_1, x_2, x_3, x_4) = (1, 1.667, 2.667, 0)$ with the objective value 22.333.



The solution $(x_1, x_2, x_3, x_4) = (1, 1.667, 2.667, 0)$ provided by the linear program at node 1 is not feasible to the integer program we want to solve because the decision variables x_2 and x_3 take fractional values in this solution, but our integer program imposes integrality constraints on these decision variables. We choose one of these decision variables, say x_2. We have $x_2 = 1.667$ in the solution at node 1, but in the optimal solution to the integer program, we must have either $x_2 \leq 1$ and $x_2 \geq 2$. Thus, at node 1, we branch into two cases, $x_2 \leq 1$ and $x_2 \geq 2$. Branching into these two cases at node 1 gives us the linear programs at nodes 3 and 4 shown in the figure below. The linear program at node 3 includes all of the constraints in the linear program at node 1, plus the constraint $x_2 \leq 1$. The linear program at node 4 includes all of the constraints in the linear program at node 1, plus the constraint $x_2 \geq 2$. In other words, the linear program at node 3 includes all of the constraints in the linear program at node 0, along with the constraints $x_1 \leq 1$ and $x_2 \leq 1$. The linear program at node 4 includes all of the constraint in the linear program at node 0, along with the constraints $x_1 \leq 1$ and $x_2 \geq 2$.



If node i lies immediately below node j, then we say that node i is a child of node j. If node j lies immediately above node i, then we say that node j is the parent of node i. For example, node 3 and node 4 in the figure above are the children of node 1 and node 1 is the

parent of node 3 and node 4. An important observation is that the optimal objective value of the linear program at a particular node is no larger than the optimal objective value of the linear program at its parent node. This observation holds because the linear program at a particular node includes all of the constraints in the linear program at its parent node, plus one more constraint. Thus, the linear program at a particular node has more constraints than the linear program at its parent node, which implies that the optimal objective value of the linear program at a particular node must be no larger than the optimal objective value of the linear program at its parent node. For example, the optimal objective value of the linear program at node 3 must be no larger than the optimal objective value of the linear program at its parent node, which is node 1.

At this point, the linear programs at nodes 2, 3 and 4 are yet unsolved. Note that the nodes we constructed so far form a tree. When choosing the next linear program to solve, we use the depth-first strategy. In other words, when choosing the next linear program to solve, we choose the deepest linear program in the tree that is yet unsolved. We discuss other options for choosing the next linear program to solve later in this chapter. Following the depth-first strategy, we need to solve the linear program at node 3 or node 4. Breaking the tie arbitrarily, we solve the linear program at node 3. Solving the linear program at node 3, we obtain the optimal objective value of 22.2 and the optimal solution is

$$x_1 = 1,\ x_2 = 1,\ x_3 = 3,\ x_4 = 0.4.$$

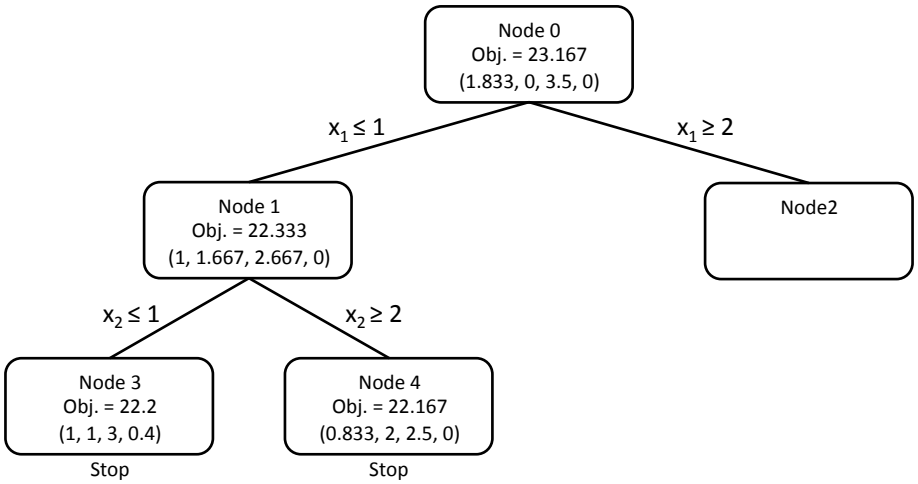
The decision variables x_1, x_2 and x_3 take integer values in this solution. So, this solution is feasible to the integer program we want to solve. Thus, we obtained a feasible solution to the integer program providing an objective value of 22.2. There is no need to explore any child nodes of node 3 further, since by the argument in the previous paragraph, the linear programs at the children of node 3 will give us objective values that are no better than 22.2 and we already have a solution to the integer program that provides an objective value of 22.2. Therefore, we can stop exploring the tree further below node 3. The best feasible solution we found so far for the integer program provides an objective value of 22.2.

The linear programs at nodes 2 and 4 are yet unsolved. Following the depth-first strategy, we solve the linear program at node 4. Solving the linear program at node 4, we obtain the optimal objective value of 22.167 and the optimal solution is

$$x_1 = 0.833,\ x_2 = 2,\ x_3 = 2.5,\ x_4 = 0.$$

The solution above is not feasible to the integer program we want to solve, because x_1 and x_3 take fractional values in this solution, whereas the integer program we want to solve requires these decision variables to be integer. However, the key observation is that the optimal objective value of the linear program at node 4 is 22.167. We know that if we explore the tree further below node 4, then the linear programs at the children of node 4 will give us objective values that are no better than 22.167. On the other hand, we already

have a feasible solution to the integer program that provides an objective value of 22.2! Recall that this solution was obtained at node 3. So, we have no hope of finding a better solution by exploring the children of node 4, which means that we can stop searching the tree further below node 4. This reasoning to stop the search at node 4 is critical for the success of the branch-and-bound method. In particular, if we have a good feasible solution to the integer program providing a high objective value, then we can stop the search at many nodes since the objective value provided by the linear program at a node would likely be worse than the objective value provided by the feasible solution to the integer program we have on hand. Being able to stop the search at many nodes would speed up the progress of the branch-and-bound method significantly. We show our progress so far in the figure below. Note that we decided to stop exploring the children of nodes 3 and 4.



The moral of the discussion in this section is that we can stop the search at the current node for two reasons. First, if the linear program at the current node provides a feasible solution to the integer program we want to solve, satisfying all integrality requirements, then we can stop the search at the current node. Second, as our search proceeds, we keep the best feasible solution to the integer program we have found so far. If the optimal objective value of the linear program at the current node is worse than the objective value provided by the best feasible solution we have found so far, then we can stop the search at the current node. Recall that the best feasible solution we have found so far for the integer program provides an objective value of 22. In the figure above, only the linear program at node 2 is unsolved. We explore node 2 and its children in the next section.

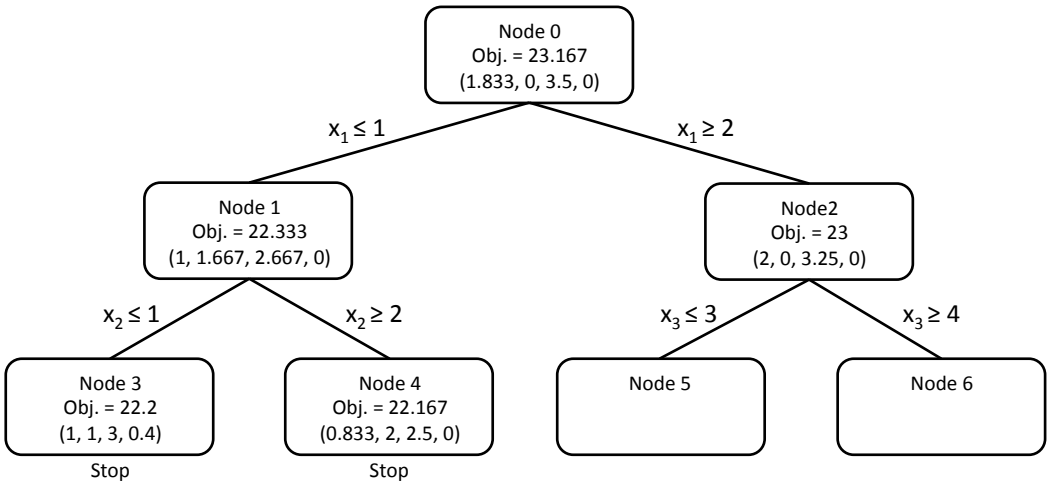
2 Another Reason to Stop the Search

Solving the linear program at node 2 in the last figure of the previous section, we obtain the optimal solution

$$x_1 = 2,\ x_2 = 0,\ x_3 = 3.25,\ x_4 = 0.$$

with the corresponding optimal objective value of 23. Note that the solution above does not satisfy the integrality requirements of the integer program we want to solve. Furthermore, the best feasible solution to the integer program we have found so far provides an objective value of 22. The optimal objective value of the linear program at node 2 is 23, which is not worse than the objective value provided by the best feasible solution we have found so far. Therefore, none of the two reasons at the end of previous section is satisfied, which implies that we do not have a reason to stop the search at node 2. So, we proceed to exploring the children of node 2.

The solution $(x_1, x_2, x_3, x_4) = (2, 0, 3.25, 0)$ provided by the linear program at node 2 is not feasible to the integer program we want to solve because the decision variable x_3 takes the fractional value 3.25 in this solution. Based on this solution at node 2, we branch into two cases, $x_3 \leq 3$ and $x_3 \geq 4$. Branching into these two cases at node 2 gives us the linear programs at nodes 5 and 6 shown in the figure below. The linear program program at node 5 includes all of the constraints in the linear program at node 2, plus the constraint $x_3 \leq 3$. The linear program at node 6 includes all of the constraints in the linear program at node 2, plus the constraint $x_3 \geq 4$. In other words, the linear program at node 5 includes all of the constraints in the linear program at node 0, along with the constraints $x_1 \geq 2$ and $x_3 \leq 3$. The linear program at node 6 includes all of the constraint in the linear program at node 0, along with the constraint $x_1 \geq 2$ and $x_3 \geq 4$.

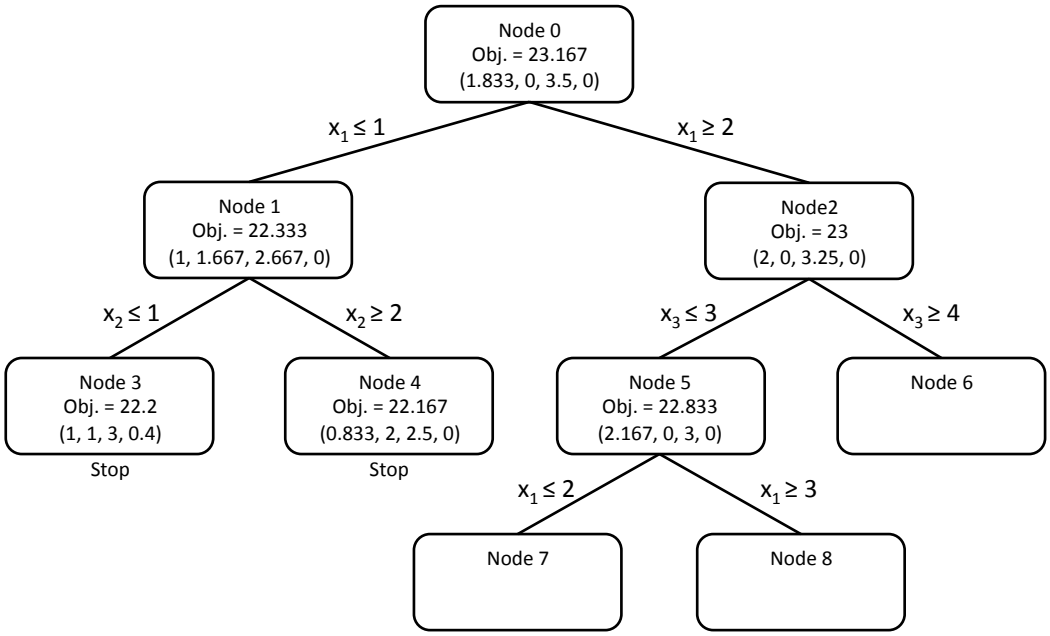


Following the depth-first strategy, we need to solve the linear program either at node 5 or at node 6. Breaking the tie arbitrarily, we proceed to solving the linear program at node 5. The optimal solution to the linear program at node 5 is

$$x_1 = 2.167,\ x_2 = 0,\ x_3 = 3,\ x_4 = 0$$

with the corresponding optimal objective value 22.833. This solution does not satisfy the integrality requirements in the integer program we want to solve. Furthermore, the optimal

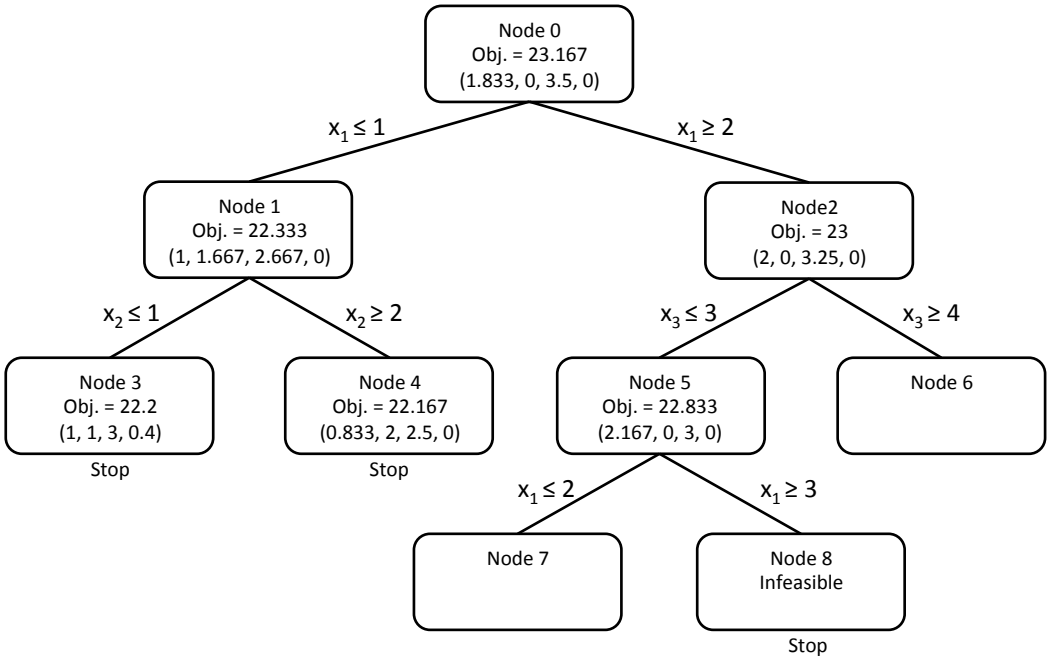
objective value of the linear program at node 5 is not worse than the objective value provided by the best feasible solution to the integer program we have found so far. Thus, we have no reason to stop the search at node 5 and we continue exploring the children of node 5. The solution $(x_1, x_2, x_3, x_4) = (2.167, 0, 3, 0)$ provided by the linear program at node 5 is not feasible to the integer program we want to solve because the decision variable x_1 takes a fractional value in this solution. Based on the solution at node 5, we branch into two cases, $x_1 \leq 2$ and $x_1 \geq 3$. Branching into these two cases at node 5 gives us the linear programs at nodes 7 and 8 shown in the figure below. The linear program at node 7 includes all of the constraints in the linear program at node 5, along with the constraint $x_1 \leq 2$. The linear program at node 8 includes all of the constraints in the linear program at node 5, along with the constraint $x_1 \geq 3$. Note that we branched into the case $x_1 \geq 2$ right before node 2. Right before node 7, we branch into the case $x_1 \leq 2$. Thus, the linear program at node 7 in effect fixes the value of x_1 at the value 2.



Now, the linear programs at nodes 6, 7 and 8 are yet unsolved. Following the depth-first strategy, we need to solve the linear program either at node 7 or node 8. Breaking the tie arbitrarily, we choose to solve the linear program at node 8. Note that the linear program at node 8 includes all of the constraints in the linear program at node 5, along with the constraint $x_1 \geq 3$. Solving the linear program at node 8, we find out that this linear program is infeasible. Since the linear programs at the children of node 8 will include all of the constraints in the linear program at node 8, the linear programs at the children of node 8 will also be infeasible. Thus, we can stop searching the tree further below node 8.

At the end of the previous section, we discussed two reasons for stopping the search at a particular node. First, if the linear program at the current node provides a solution that

satisfies the integrality requirements in the integer program we want to solve, then we can stop the search at the current node. Second, if the optimal objective value of the linear program at the current node is worse than the objective value provided by the best feasible solution to the integer program we have found so far, then we can stop the search at the current node. The discussion in this section provides a third reason to stop the search at a particular node. If the linear program at the current node is infeasible, then we can stop the search at the current node. We summarize our progress so far in the figure below.



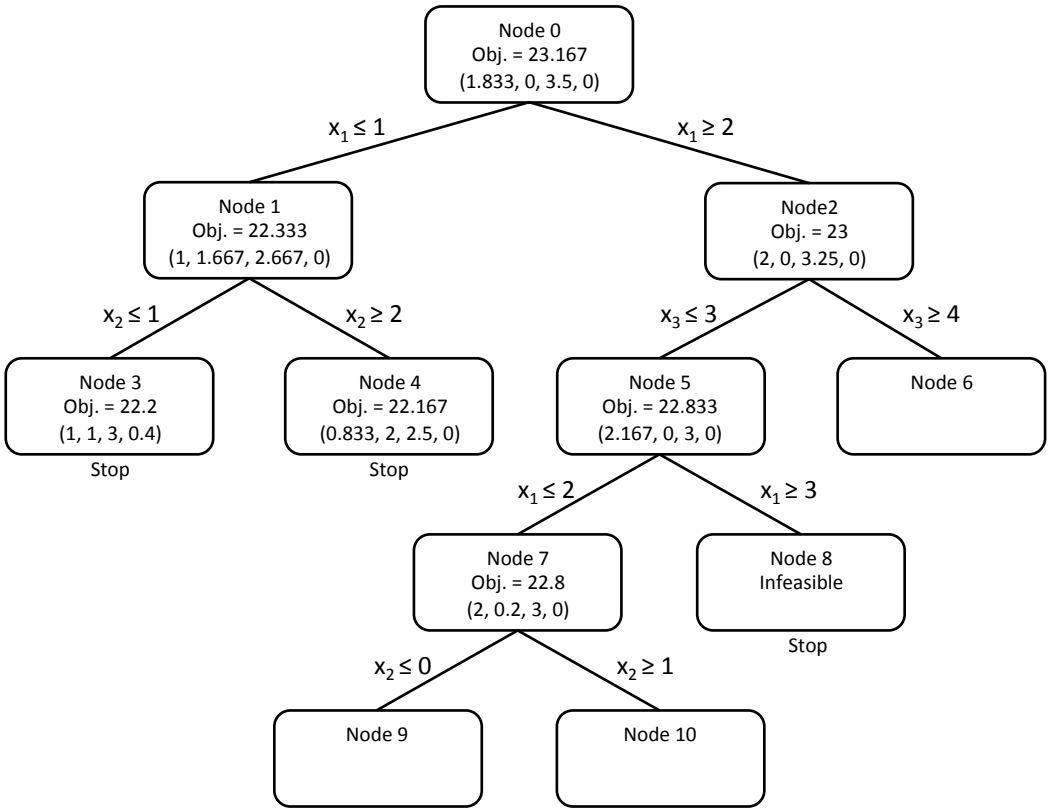
3 Completing the Branch-and-Bound Method

In the last figure of the previous section, the linear programs at nodes 6 and 7 are yet unsolved. Following the depth-first strategy, we solve the linear program at node 7. The solution to the linear program at node 7 is

$$x_1 = 2,\ x_2 = 0.2,\ x_3 = 3,\ x_4 = 0$$

with the corresponding optimal objective value 22.8. The solution above does not satisfy the integrality requirements in the integer program we want to solve. Also, the optimal objective value of the linear program at node 7 is not worse than the objective value provided by the best feasible solution to the integer program we have found so far. So, we have no reason to stop the search at node 7. The decision variable x_2 needs to take an integer value in the integer program we want to solve, but we have $x_2 = 0.2$ in the solution to the linear program at node 7. Based on the solution of the linear program at node 7, we branch into the cases $x_2 \leq 0$ and $x_2 \geq 1$. Branching into these cases yields the linear programs at nodes 9 and 10 shown in the figure below. The linear program at node 9 includes all of the constraints

in the linear program at node 7 and the constraint $x_2 \leq 0$. The linear program at node 10 includes all of the constraints in the linear program at node 7 and the constraint $x_2 \geq 1$.



Now, the linear programs at nodes 6, 9 and 10 are unsolved. By the depth-first strategy, we solve the linear program at node 9 or node 10. Breaking the tie arbitrarily, we solve the linear program at node 9. The optimal solution to the linear program at node 9 is

$$x_1 = 2,\ x_2 = 0,\ x_3 = 3,\ x_4 = 0.2$$

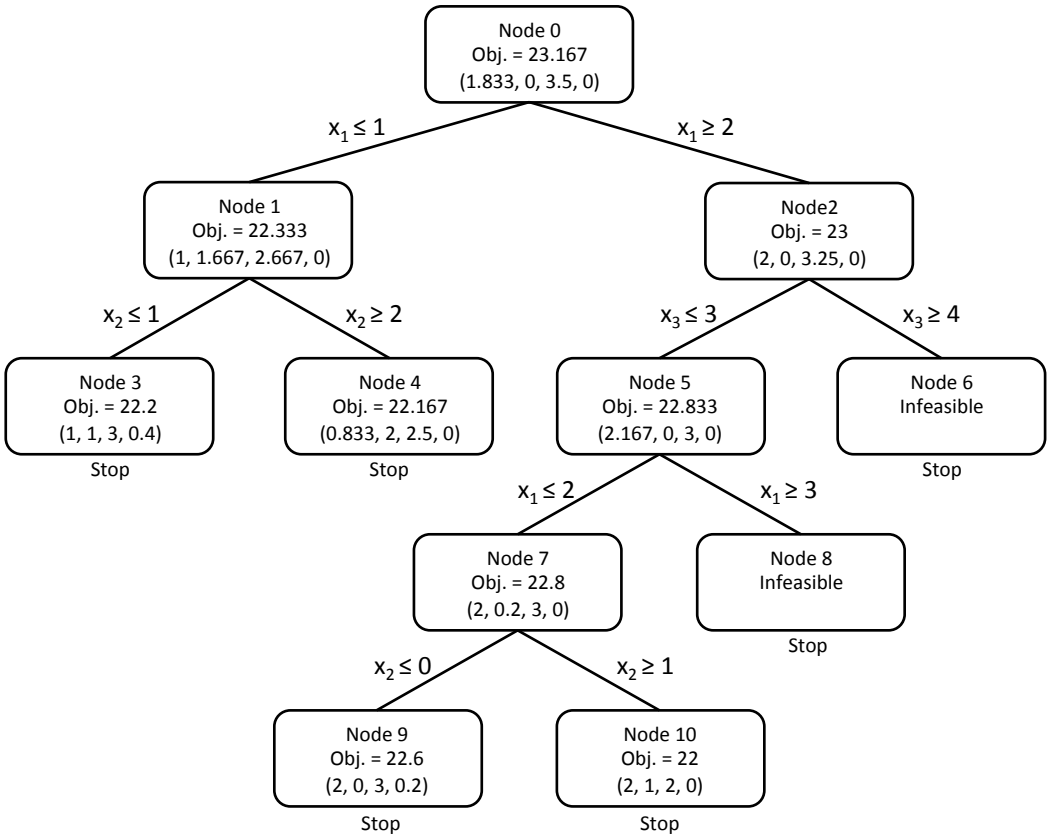
with the corresponding optimal objective value 22.6. This solution satisfies all of the integrality requirements in the integer program we want to solve. So, we do not need to explore the children of node 9. The solution provided by the linear program at node 9 is a feasible solution to the integer program we want to solve. Before node 9, the best feasible solution we had for the integer program provided an objective value of 22. However, the solution that we obtained at node 9 is feasible to the integer program we want to solve and it provides an objective value of 22.6. Thus, we update the best feasible solution we have found so far as the solution obtained at node 9.

At this point, the linear programs at nodes 6 and 10 are unsolved. Following the depth-first strategy, we solve the linear program at node 10. The optimal objective value of this linear program is 22 and the optimal solution is

$$x_1 = 2,\ x_2 = 1,\ x_3 = 2,\ x_4 = 0.$$

This solution satisfies all of the integrality requirements in the integer program we want to solve. Therefore, there is no reason to explore the children of node 10. We can stop searching the tree below node 10.

The only unsolved linear program left is at node 6. Solving this linear program, we see that the linear program at node 6 is infeasible. Thus, there is no reason to explore the children of node 6. The figure below shows our current progress.



There are no unsolved linear programs in the figure above. So, our search is complete! The best feasible solution to the integer program is the solution that we obtained at node 9. Therefore, we can conclude that the solution $(x_1, x_2, x_3, x_4) = (2, 0, 3, 0.2)$ is optimal to the integer program we want to solve.

4 Summary of the Branch-and-Bound Method

It is worthwhile to summarize some of the important points about the branch-and-bound method. As our search over the tree progresses, we keep track of the best feasible solution to the integer program we have found so far. After solving the linear program at the current node, we can stop the search at the current node for one of three reasons.

- The solution to the linear program at the current node provides a feasible solution

to the integer program we want to solve, satisfying all integrality requirements in the integer program.

- The optimal objective value of the linear program at the current node is worse than the objective value provided by the best feasible solution to the integer program we have found so far.
- The linear program at the current node is infeasible.

If none of the three reasons above hold and we cannot stop the search at the current node, then we branch into two cases, yielding two more linear programs to solve. The second reason above is critical to the success of the branch-and-bound method. In particular, if we have a good feasible solution to the integer program on hand, then the optimal objective value of the linear program at the current node is more likely to be worse than the objective value provided by the feasible solution we have on hand. Thus, we can immediately terminate the search at the current node. The good feasible solution to the integer program we have on hand could either be obtained during the course of the search in the branch-and-bound method or be obtained by using a separate heuristic solution algorithm.

Throughout this chapter, we used the depth-first strategy when selecting the next linear program to solve. The advantage of the depth-first strategy is that it allows us to obtain a feasible solution to the integer program quickly. In particular, the nodes towards the beginning of the tree do not have many constraints added in them. Thus, they are less likely to provide feasible solutions satisfying the integrality requirements in the integer program we want to solve. On the other hand, the nodes towards the bottom of the tree have many constraints added in them and they are likely to provide solutions that satisfy the integrality requirements. As discussed in the previous paragraph, having a good feasible solution on hand is critical to the success of the branch-and-bound method. Another approach for selecting the next linear program to solve is to focus on the node that includes the linear program with the largest optimal objective value and solve the linear program corresponding to one of its children.

After solving the linear program at a particular node, there may be several variables that violate the integrality requirements of the integer program we are interested in solving. In this case, we can use any one of these decision variables to branch on. For example, if the decision variables x_1 and x_2 are restricted to be integers, but we have $x_1 = 2.5$ and $x_2 = 4.7$ in the optimal solution to the linear program at the current node, then we two options for the decision variable to branch on. First, we can branch on the decision variable x_1 and use the two cases $x_1 \leq 2$ and $x_1 \geq 3$ to construct the child nodes of the current node. Second, we can branch on the decision variable x_2 and use the two cases $x_2 \leq 4$ and $x_2 \geq 5$ to construct the child nodes of the current node. The choice of a good variable to branch on is hard to figure out a priori, but choosing a good variable to branch on may have dramatic impact on the size of the search tree. A general rule of thumb is that if there is some hierarchical ordering

between the decisions, then one should first branch on the decision variables that represent higher order decisions. For example, if we have decision variables on which facilities to open and decision variables on which demand points the open facilities should serve, then we should probably first branch on the decision variables that represent which facilities to open. Nevertheless, in many practical decision-making problems, it is hard to see a hierarchical ordering between the decisions and one branching strategy that works well in one problem setting may not work well in other settings. The choice of the next node to focus on and the choice of the decision variable to branch on are two of the reasons that make integer programs substantially more difficult to solve than linear programs.

Modeling in Logistics

Numerous problems in the field of logistics can be formulated either as linear programs or as integer programs. In this chapter, we discuss uses of linear and integer programs for modeling problems in logistics.

1 Facility Location Problem

We want to locate facilities to serve the demand at a number of demand points scattered over a geographical region. The set of possible locations for the facilities is F. The set of demand points is D. If we open a facility at location j, then we incur a fixed cost of f_j. The cost of serving demand point i from a facility at location j is c_{ij}. Each demand point must be served from one facility. We want to figure out where to open facilities and which facilities to use to serve each demand point to minimize the total cost of opening the facilities and serving the demand points. The data for the problem are the set F of possible locations for the facilities, the set D of demand points, the fixed costs $\{f_j : j \in F\}$ of opening facilities at different locations and the costs $\{c_{ij} : i \in D,\ j \in F\}$ of serving different demand points from facilities at different locations. To formulate the problem as an integer program, we make use of the decision variables

$$x_j = \begin{cases} 1 & \text{if we open a facility at location } j \\ 0 & \text{otherwise,} \end{cases}$$

$$y_{ij} = \begin{cases} 1 & \text{if we serve demand point } i \text{ from a facility at location } j \\ 0 & \text{otherwise.} \end{cases}$$

Note that if $x_j = 0$, meaning that we do not have a facility at location j, then we cannot serve demand point i from a facility at location j, meaning that we must have $y_{ij} = 0$. To capture this relationship between the decision variables x_j and y_{ij}, we use the constraint $y_{ij} \leq x_j$. Thus, to choose the locations for facilities and to decide which facilities to use to serve each demand point, we can solve the integer program

The objective function accounts for the total cost of opening the facilities and serving the

demand points. Noting the definition of the decision variable y_{ij} above, $\sum_{j \in F} y_{ij}$ in the first constraint corresponds to the number of facilities that serve demand point i. Thus, the first constraint ensures that each demand point i is served by one facility. The second constraint ensures that if we do not have a facility at location j, then we cannot use a facility at location j to serve demand point i. The problem above is known as the uncapacitated facility location problem. In particular, our formulation assumes that as long as we have a facility at a certain location, we can serve as many demand points as we like from that location. So, our formulation of the facility location problem assumes that there is infinite capacity at the facilities. That is, the facilities are uncapacitated.

There is a capacitated version of the facility location problem. The setup for the capacitated facility location is the same as before. The only difference is that demand point i has a demand of d_i units. The total demand served by any facility cannot exceed U. Similar to our formulation of the uncapacitated facility location problem, we continue assuming that each demand point is served by one facility. We want to figure out the locations for facilities and the facilities used to serve each demand point, while making sure that the total demand served by a facility does not exceed the capacity at the facility. This problem can be formulated as the integer program

The objective function and the first constraint are identical in the uncapacitated and capacitated facility location problems. If we have $x_j = 0$, then the second constraint above reads $\sum_{i \in D} d_i \, y_{ij} \leq 0$. To satisfy this constraint, we must set $y_{ij} = 0$ for all $i \in D$. Thus, if we have $x_j = 0$, meaning that we do not have a facility at location j, then we must have $y_{ij} = 0$ for all $i \in D$, meaning that no demand point can be served from a facility at location j. If we have $x_j = 1$, then the second constraint above reads $\sum_{i \in D} d_i \, y_{ij} \leq U$. Note that $\sum_{i \in D} d_i \, y_{ij}$ is the total demand at the demand points served by the facility at location j. Thus, if we have $x_j = 1$, meaning that we have a facility at location j, then we must have $\sum_{i \in D} d_i \, y_{ij} \leq U$, meaning that the total demand at the demand points served by the facility at location j must be no larger than the capacity of the facility.

In our formulation of the capacitated facility location problem above, we could add the constraints $y_{ij} \leq x_j$ for all $i \in D$, $j \in F$. These constraints would be redundant because

the constraint $\sum_{i \in D} d_i \, y_{ij} \leq U \, x_j$ already ensures that if a facility is not open at location j, then we cannot serve any demand point from a facility at location j. Thus, the optimal objective value of the capacitated facility location problem would not change when we add the constraints $y_{ij} \leq x_j$ for all $i \in D$, $j \in F$. However, the optimal objective value of the linear programming relaxation of the capacitated facility location problem could change when we add the constraints $y_{ij} \leq x_j$ for all $i \in D$, $j \in F$. So, although adding these constraints increases the number of constraints in the formulation, there can be some value adding these constraints into the formulation, because practical solvers such as Gurobi use linear programming relaxations when solving the problem through the branch-and-bound method. Adding these constraints into the formulation may help the branch-and-bound method obtain integer solutions substantially faster.

2 Dynamic Driver Assignment Problem

We are managing drivers in a transportation network during the course of T days. The set of locations in the transportation network is N. At the beginning of day 1, we have s_i drivers at location i. On day t, we have d_{ijt} loads available that should be carried from location i to j. To carry a load from location i to j on day t, we must have a driver available at location i on day t. Each driver carries one load at a time. The travel time between each pair of locations is a single day. In particular, if a driver at location i at the beginning of day t carries a load to location j, then he becomes available at location j at the beginning of day $t + 1$. We have the option of letting a driver stay at his current location. If a driver at location i stays at this location on day t, then this driver is available at location i at the beginning of day $t+1$ to carry a load. If we carry a load from location i to j, then we generate a revenue of r_{ij}. We want to figure out how many loads to carry between each location pair on each day to maximize the total revenue. The data for the problem are the number T of days in the planning horizon, the set N of locations, the numbers $\{s_i : i \in N\}$ of drivers at different locations at the beginning of day 1, the numbers $\{d_{ijt} : i, j \in N, \ t = 1, \ldots, T\}$ of loads to be carried between different location pairs on different days and the associated revenues $\{r_{ij} : i, j \in N\}$. We use the following decision variables.

x_{ijt} = Number of drivers that carry a load from location i to j on day t.

z_{it} = Number of drivers that stay at location i on day t.

To decide which loads to carry during the course of T days, we can solve the problem

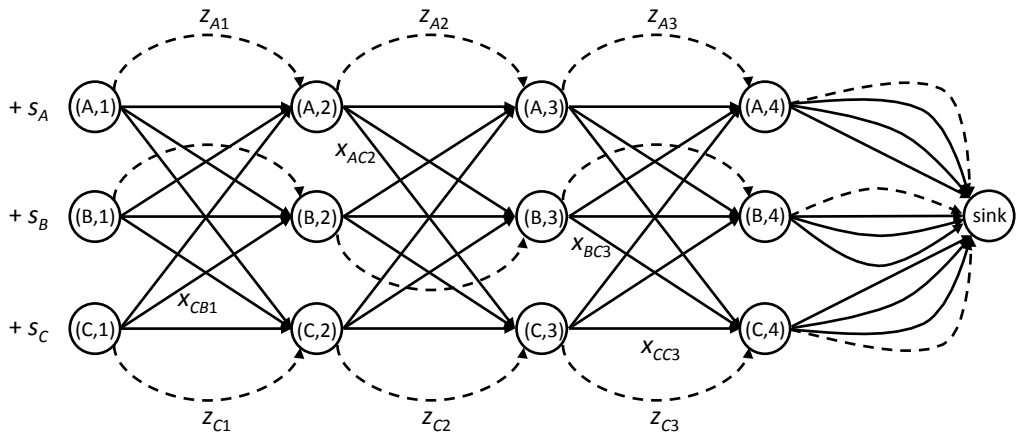
The objective function accounts for the total revenue collected from the loads carried between all location pairs and on all days. In the first constraint, $\sum_{j \in N} x_{ij1}$ corresponds to the number of drivers leaving location i on day 1. Thus, $\sum_{j \in N} x_{ij1} + z_{i1}$ on the left side of the first constraint corresponds to the total number of drivers leaving location i or staying at location i on day 1. In this case, the first constraint ensures that the total number of number of drivers leaving location i or staying at location i on day 1 should be equal to the number of drivers available at location i at the beginning of day 1.

Similarly, $\sum_{j \in N} x_{ijt} + z_{it}$ on the left side of the second constraint corresponds to the total number of drivers leaving location i or staying at location i on day t. On the other hand, $\sum_{j \in N} x_{ji,t-1}$ in the second constraint corresponds to the number of drivers that started moving towards location i on day $t-1$. These drivers will be available at location i at the beginning of day t. Similarly, $z_{i,t-1}$ is the number of drivers that stay at location on day $t-1$. These drivers will be available at location i at the beginning of day t as well. Thus, $\sum_{j \in N} x_{ji,t-1} + z_{i,t-1}$ on the right side of the second constraint gives the total number of drivers that are available at location i at the beginning of day t. In this case, the second constraint ensures that the total number of number of drivers leaving location i or staying at location i on day t should be equal to the total number of drivers available at location i at the beginning of day t. The third set of constraints ensures that the number of drivers that carry a load from location i to j on day t cannot exceed the number of loads between this location pair on this day. We observe that our formulation of the problem assumes that if a load that needs to be carried on day t cannot be carried on that day, then the load is lost. In particular, the load cannot be carried on a future day. Also, our formulation assumes that there can be loads that need to be carried from location i to location i.

We will refer to the problem above as the dynamic driver assignment problem. There are a few important lessons to derive from our formulation of the dynamic driver assignment problem. This formulation captures a problem that takes place over time. An important

approach for formulating problems that takes place over time is to create copies of the decision variables that correspond to the decisions made at different time periods. For example, we have a decision variable x_{ijt} for each day t that captures the number of drivers that carry a load from location i to j on each day. The objective function accounts for the reward or the cost over the whole planning horizon as a function of the decisions over the whole planning horizon. We have some constraints that capture the relationship between the decisions made at different time periods. For example, the drivers that carry loads and that stay at their current locations on day $t-1$ dictate the numbers of drivers available at different locations at the beginning of day t. We capture this relationship by using the second set of constraints in our formulation of the dynamic driver assignment problem. There are also constraints on the decisions made at each time period. For example, the number of drivers that carry a load from location i to j on day t cannot exceed the number of loads available between this location pair on day t. We capture this constraint by the third set of constraints in our formulation. The idea of dividing a planning horizon into a number of time periods and creating copies of the decision variables that capture the decisions made at different time periods plays a crucial role in many optimization models used in practice today. In the dynamic driver assignment problem, we divided the planning horizon into days, but if the decisions are made more frequently than once per day, then we can divide the planning horizon into 4-hour time periods, hours or even minutes!

Another important point about our formulation of the dynamic driver assignment problem is that it corresponds to a min-cost network flow problem taking place over a special network. Consider the network in the figure below. In this network, we assume that the set of locations is $N = \{A, B, C\}$ and the number of days is $T = 4$. We have one node for each location-day pair. Therefore, we can index the nodes by (i, t), where $i \in N$ and $t = 1, \ldots, T$. For days $t = 1, \ldots, T-1$, the decision variable x_{ijt} corresponds to the flow on an arc from node (i, t) to node $(j, t+1)$. The flow on this arc corresponds to the number of drivers that carry a load from location i to j on day t. These drivers become available at location j at the beginning of day $t+1$. For days $t = 1, \ldots, T-1$, the decision variable z_{it} corresponds to the flow on an arc from node (i, t) to node $(i, t+1)$. The flow on this arc corresponds to the number of drivers that we keep at location i on day t. These drivers become available at location i at the beginning of day $t+1$. For the last day T, the decision variable x_{ijT} corresponds to the flow on an arc from node (i, T) to the special sink node. The flow on this arc corresponds to the number of drivers that carry a load from location i to j on day T. Since the planning horizon ends on day T, we do not need to worry about the destinations of the drivers that carry loads on day T. Thus, the arcs on day T all terminate at the same sink node. Similarly, the decision variable z_{iT} corresponds to the flow on an arc from node (i, T) to the sink node. In the figure below, the arcs corresponding to the decision variables $\{x_{ijt} : i, j \in N,\ t = 1, \ldots, T\}$ are in solid lines and the arcs corresponding to the decision variables $\{z_{it} : i \in N,\ t = 1, \ldots, T\}$ are in dashed lines.



Noting the discussion in the previous paragraph, the decision variable x_{ijt} corresponds to the flow on an arc that goes from node (i,t) to $(j,t+1)$. The decision variable z_{it} corresponds to the flow on an arc that goes from node (i,t) to $(i,t+1)$. Thus, the total flow out of node (i,t) is $\sum_{j \in N} x_{ijt} + z_{it}$. On the other hand, the decision variable $x_{ji,t-1}$ corresponds to the flow on an arc that goes from node $(j,t-1)$ to (i,t). Similarly, the decision variable $z_{i,t-1}$ corresponds to the flow on an arc that goes from node $(i,t-1)$ to (i,t). Thus, the total flow into node (i,t) is $\sum_{j \in N} x_{ji,t-1} + z_{i,t-1}$. Therefore, the second set of constraints in the dynamic driver assignment problem captures the flow balance constraints for the node (i,t) for all $i \in N$ and $t = 2, \ldots, T$. The node $(i,1)$ does not have any incoming arcs, but the node $(i,1)$ has a supply of s_i units, which is the number of drivers available at node i at the beginning of day 1. Thus, the first set of constraints in the dynamic driver assignment problem captures the flow balance constraints for the node $(i,1)$ for all $i \in N$. We have an upper bound of d_{ijt} on the flow over the arc corresponding to the decision variable x_{ijt}. Our formulation of the dynamic driver assignment problem does not include a flow balance constraint for the sink node in the figure above, but we know that in a min-cost network flow problem, the flow balance constraint of one node is always redundant. Thus, our formulation of the dynamic driver assignment problem omits the flow balance constraint for the sink node. Lastly, the dynamic driver assignment problem maximizes its objective function rather than minimizing as in a min-cost network flow problem, but we can always minimize the negative of the objective function in the dynamic driver assignment problem. Thus, the dynamic driver assignment problem corresponds to a min-cost network flow problem over the network shown above with upper bounds on the flows over some of the arcs.

Recall that if all of the demand and supply data in a min-cost network flow problem are integer-valued, then there exists an integer-valued optimal solution even when we do not impose integrality requirements on the decision variables. It turns out this result continues to hold when we have upper bounds on the flows over some of the arcs and these upper bounds are also integer-valued. Therefore, it follows that if the numbers of drivers at different locations at the beginning of day 1 are integers and the numbers of loads that need to be

carried between different location pairs on different days are integers, then there exists an integer-valued optimal solution to the dynamic driver assignment problem even when we do not impose the integrality requirements on the decision variables. In this case, we can drop all of the integrality constraints to solve the linear programming relaxation of the dynamic driver assignment problem and still get an integer-valued optimal solution.

The network in the figure above is called a state-time network, where the state captures the locations of the drivers and the time captures the different days in the planning horizon. State-time networks are powerful tools for modeling logistics problems. They have been successfully used in freight applications as discussed in this section. State-time networks also play an important role in optimization models that airlines use when assigning aircraft to flights. When assigning aircraft to flights, the state corresponds to the location of an aircraft and the time corresponds to the departure and arrival times of the flights.

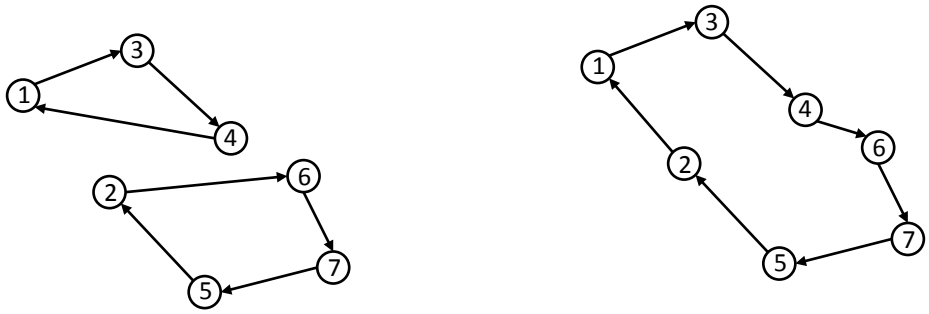
3 Traveling Salesman Problem

We have a set N of cities. There is an arc between every pair of cities. We denote the arc from city i to city j as arc (i, j). The distance associated with arc (i, j) is c_{ij}. Starting from one of the cities, we want to find a tour of cities with minimum total distance such that the tour travels each city exactly once and returns back to the starting city. This problem is known as the traveling salesman problem. To formulate the traveling salesman problem as an integer program, we use the decision variable

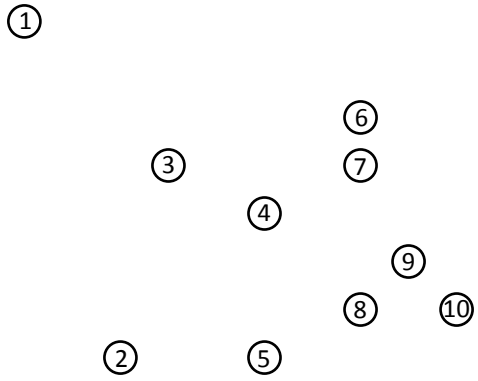
$$x_{ij} = \begin{cases} 1 & \text{if arc } (i,j) \text{ is included in the tour} \\ 0 & \text{otherwise.} \end{cases}$$

In an acceptable tour, we must depart each city i exactly once. In other words, we must use exactly one of the arcs that go out of each city i. We can represent this requirement by using the constraint $\sum_{j \in N} x_{ij} = 1$. Similarly, we must enter each city i exactly once. So, we must use exactly one of the arcs that go into each city i. This requirement can be represented by using the constraint $\sum_{j \in N} x_{ji} = 1$. In this case, we can formulate the traveling salesman problem as the integer program

In the objective function, we account for the total distance of the arcs included in the tour. The first constraint ensures that we depart each city exactly once, whereas the second constraint ensures that we enter each city exactly once. It turns out these two sets of constraints are not adequate to find an acceptable tour. For example, consider the 7 cities in the figure below. In the tour on the left side of the figure, we depart each city exactly once and we enter each city exactly once, but the solution is not a single tour that starts from one of the cities and ends at the same starting city. In particular, there are subtours in the solution. The third set of constraints above is known as subtour elimination constraints. The subtour elimination constraints state that if we partition the cities into two subsets S and $N \setminus S$, then to avoid having subtours in the solution, we must use at least one arc that directly connects a city in set S to a city in set $N \setminus S$. That is, any partition of the cities should be connected to each other. Otherwise, the solution would include subtours. For example, the tour on the left side of the figure below includes subtours because if we partition the cities into the sets $S = \{1, 3, 4\}$ and $N \setminus S = \{2, 5, 6, 7\}$, then the tour in the figure does not use an arc that directly connects a city in set S to a city in $N \setminus S$. As a result, the tour includes subtours. The tour on the right side of the figure below does not include any subtours because if we partition the cities into any two sets S and $N \setminus S$, then the tour on the right side always includes an arc that directly connects a city in S to a city in $N \setminus S$. For example, the tour on the right side of the figure below does include an arc that directly connects a city in the set $S = \{1, 3, 4\}$ and to a city in the set $N \setminus S = \{2, 5, 6, 7\}$, which is arc $(4, 6)$. As a minor detail, note that our formulation includes a decision variable x_{ii} for each city i, which implies that there is an arc that goes from city i back to city i. We can set the cost c_{ii} of this arc large so that this arc is never used in the optimal solution.



We have one subtour elimination constraint for each subset of the cities. Thus, if there are n cities, then there are 2^n subtour elimination constraints, which can easily get large. With this many constraints, our formulation of the traveling salesman problem appears to be useless! The trick to using our formulation is to add the subtour elimination constraints as needed. To illustrate the idea, consider the 10 cities on the left side of the figure below. On the right side, we show the distance from city i to city j for all $i, j \in N$.



i \ j	1	2	3	4	5	6	7	8	9	10
1	·	7	4	6	9	7	8	9	9	11
2	7	·	4	4	3	7	6	5	6	7
3	4	4	·	2	4	4	4	5	5	7
4	6	4	2	·	3	3	2	3	3	4
5	9	3	4	3	·	5	4	2	4	4
6	7	7	4	3	5	·	1	4	3	4
7	8	6	4	2	4	1	·	3	2	4
8	9	5	5	3	2	4	3	·	1	2
9	9	6	5	3	4	3	2	1	·	1
10	11	7	7	4	4	4	4	2	1	·

We begin by solving the formulation of the traveling salesman problem without any subtour elimination constraints. In particular, we minimize the objective function in the traveling salesman problem subject to the constraints $\sum_{j \in N} x_{ij} = 1$ for all $i \in N$ and $\sum_{j \in N} x_{ji} = 1$ for all $i \in N$ only. The figure below shows the optimal solution that we obtain when we solve the formulation of the traveling salesman problem without any subtour elimination constraints. In particular, we have $x_{13} = x_{31} = x_{24} = x_{48} = x_{85} = x_{52} = x_{67} = x_{76} = x_{9,10} = x_{10,9} = 1$ in the optimal solution and the other decision variables are zero.

In the solution in the figure above, we have a subtour that includes the set of cities $S = \{1, 3\}$. That is, the solution above does not include an arc that connects a city in $S = \{1, 3\}$ directly to a city in $N \setminus S = \{2, 4, 5, 6, 7, 8, 9, 10\}$. Thus, we add the subtour elimination constraint corresponding to the set $S = \{1, 3\}$ into our formulation. Note that this subtour elimination constraint is given by

$$x_{12} + x_{14} + x_{15} + x_{16} + x_{17} + x_{18} + x_{19} + x_{1,10} + x_{32} + x_{34} + x_{35} + x_{36} + x_{37} + x_{38} + x_{39} + x_{3,10} \geq 1.$$

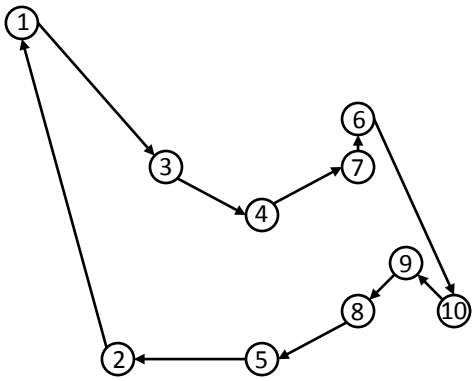
In the constraint above, the first index of the decision variables is a city in set S and the

second index of the decision variables is a city in the set $N \setminus S$. Similarly, the solution above has a subtour that includes the set of cities $S = \{2, 4, 5, 8\}$. We add the subtour elimination constraint corresponding to this set S as well. The subtour elimination constraint corresponding to the set $S = \{2, 4, 5, 8\}$ can be written as

$$x_{21} + x_{23} + x_{26} + x_{27} + x_{29} + x_{2,10} + x_{41} + x_{43} + x_{46} + x_{47} + x_{49} + x_{4,10} \\ + x_{51} + x_{53} + x_{56} + x_{57} + x_{59} + x_{5,10} + x_{81} + x_{83} + x_{86} + x_{87} + x_{89} + x_{8,10} \geq 1.$$

There is another subtour in the solution above that includes the set of cities $S = \{6, 7\}$. We add the subtour elimination constraint corresponding to this set S into our formulation. Lastly, the solution above has one more subtour that includes the set of cities $S = \{9, 10\}$. We add the subtour elimination constraint corresponding to this set of cities as well. The subtour elimination constraints corresponding to the sets $S = \{6, 7\}$ and $S = \{9, 10\}$ can be written by using an argument similar to the one used in the two subtour elimination constraints above. Therefore, we added 4 subtour elimination constraints. Solving our formulation of the traveling salesman problem with these 4 subtour elimination constraints, we obtain the solution in the figure below.

The solution in the figure above includes three subtours. Noting the cities involved in each one of these subtours, we further add the 3 subtour elimination constraints corresponding to the sets $S = \{1, 3, 4, 6, 7\}$, $S = \{2, 5\}$ and $S = \{8, 9, 10\}$ into our formulation of the traveling salesman problem. Considering the 4 subtour elimination constraints that we added earlier, we now have a total of 7 subtour elimination constraints. Solving the formulation of the traveling salesman problem with these 7 subtour elimination constraints, we obtain the solution given in the figure below.



The solution above does not include any subtours. By adding 7 subtour elimination constraints into the formulation of the traveling salesman problem, we obtained a solution that does not have any subtours. Since this solution does not include any subtours, it must be the optimal solution when we solve the traveling salesman problem with all subtour elimination constraints. Therefore, the solution shown above is the optimal solution for the traveling salesman problem. The total distance of this tour is 27. Note that the traveling salesman problem we dealt with involves 10 cities. Thus, if we constructed all of the subtour elimination constraints at the beginning, then we would have to construct $2^{10} = 1024$ subtour elimination constraints. By generating the subtour elimination constraints as needed, we were able to obtain the optimal solution to the traveling salesman problem by generating only 7 subtour elimination constraints. For a problem with 10 cities, constructing all of the 1024 subtour elimination constraints may not be difficult. However, if we have a problem with 100 cities, then there are $2^{100} \approx 10^{30}$ subtour elimination constraints and it is impossible to construct all of these subtour elimination constraints. Although it is not possible to construct all of the subtour elimination constraints, traveling salesman problems with hundreds of cities are routinely solved today. Lastly, we emphasize that the idea of adding the constraints to an optimization problem as needed is an effective approach to tackle problems with a large number of constraints. In this section, we used this approach to solve the traveling salesman problem, but we can use the same approach when dealing with other optimization problems with large numbers of constraints.

Designing Heuristics

In the previous chapter, we discussed problems in logistics that can be modeled as integer programs. In many cases, we can solve the integer programs by using available optimization software. However, when the problem gets too large or too complicated, we may have to resort to heuristic methods to obtain a solution. Heuristic methods are designed to find a good solution to the problem on hand, but they have no guarantee of finding the optimal solution. Also, during our discussion of the branch-and-bound method, we saw how a good feasible solution to the problem may allow us to stop the search process quickly at different nodes of the tree. Therefore, obtaining a good solution by using a heuristic method may also be useful when we subsequently try to obtain the optimal solution to the problem by using the branch-and-bound method.

1 Prize-Collecting Traveling Salesman Problem

To demonstrate the fundamental ideas in designing heuristics, we use the prize-collecting traveling salesman problem. We have a set N of cities. There is an arc between every pair of cities. We denote the arc from city i to city j as arc (i, j). The distance associated with arc (i, j) is c_{ij}. Associated with each city i, there is a reward of r_i. The profit from a tour of cities is given by the difference between the total reward collected at the cities visited in the tour and the total distance of the arcs included in the tour. We start our tour from a given city $0 \in N$. We are interested in finding a tour that visits a subset of the cities such that the tour starts and ends at city 0 and the profit from the tour is maximized.

To design a heuristic method to obtain a good solution to the prize-collecting traveling salesman problem, we begin by thinking about how we can denote a possible solution to the problem. We denote a solution by keeping a sequence of cities. In particular, we denote a possible solution to the problem as $(j_0, j_1, \ldots, j_n)$, where n is the number of cities in the tour, the first city j_0 in the tour is city 0 and the subsequent cities visited in the tour are $j_1, j_2, \ldots, j_n$. Since city 0 must always be visited, we choose not to count city 0 in the number of cities visited in the tour, but this choice is simply a matter of notational convention. Next, we think about how we can compute the objective value corresponding to a solution. The profit from the solution $(j_0, j_1, \ldots, j_n)$ is given by

$$f(j_0, j_1, \ldots, j_n) = r_{j_1} + \ldots + r_{j_n} - c_{j_0,j_1} - c_{j_1,j_2} - \ldots - c_{j_{n-1},j_n} - c_{j_n,j_0}.$$

In the profit expression above, we do not include a reward for city 0 because we know that this city must be visited in any tour anyway. Also, since we must go back to city 0 after visiting the last city j_n, we include the cost c_{j_n,j_0} in the profit expression above. Generally speaking, there are two classes of heuristics, construction heuristics and improvement heuristics. In the next two sections, we discuss these two classes of heuristics within the context of the prize-collecting traveling salesman problem.

2 Construction Heuristics

In a construction heuristic, we start with an empty solution. What we mean by an empty solution depends on the specific problem on hand. For the prize-collecting traveling salesman problem, an empty solution could correspond to the tour where we only visit city 0 to collect a profit of 0. In a construction heuristic, we start with an empty solution and progressively construct better and better solutions. A common idea to design a construction heuristic is to be greedy and include an additional component into the solution that provides the largest immediate increase in the objective value. In the prize-collecting traveling salesman problem, this idea could result in inserting a city into the current tour such that the inserted city provides the largest immediate increase in the profit of the current tour.

To give the details of a construction heuristic for the prize-collecting traveling salesman problem, assume that the current tour on hand is $(j_0, j_1, \ldots, j_n)$. We consider each city k that is not in the current tour. We try inserting city k into the current tour at each possible position and check the increase in the profit. We choose the city that provides the largest increase in the profit of the current tour and insert this city into the tour at the position that provides the largest increase in the profit. In particular, assume that we currently have the solution $(j_0, j_1, \ldots, j_n)$ with n cities in it. We consider a city $k \in N \setminus \{j_0, j_1, \ldots, j_n\}$ that is not in the current tour. If we add city k into the the current tour after the ℓ-th city, then the increase in the profit is given by

$$\Delta_k^\ell(j_0, j_1, \ldots, j_n) = f(j_0, j_1, \ldots, j_\ell, k, j_{\ell+1}, \ldots, j_n) - f(j_0, j_1, \ldots, j_\ell, j_{\ell+1}, \ldots, j_n).$$

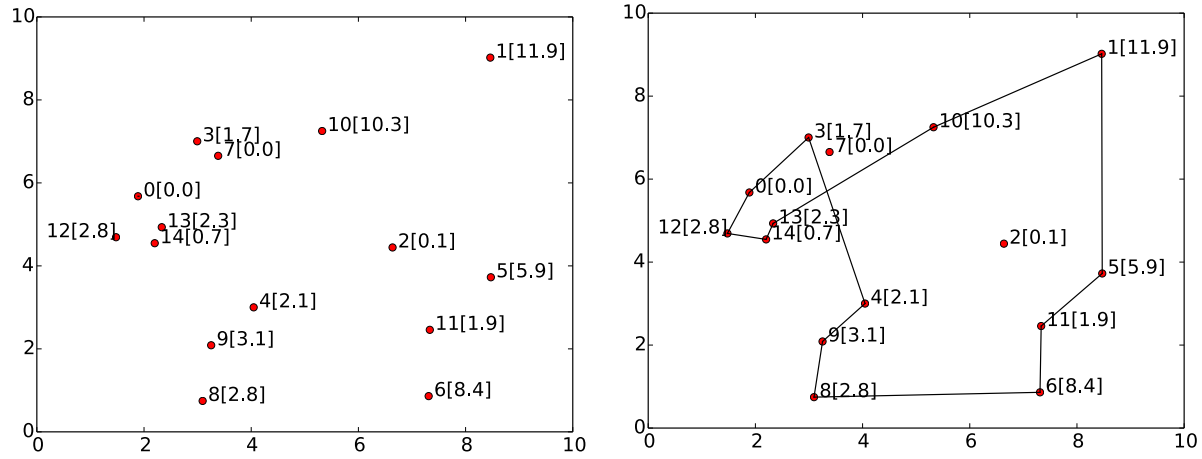
We note that the increase in the profit given above can be a negative quantity. We choose the city k^* and the position ℓ^* that maximizes the increase in the profit. That is, the city k^* and the position ℓ^* is given by

$$(k^*, \ell^*) = \arg\max\{\Delta_k^\ell(j_0, j_1, \ldots, j_n) : k \in N \setminus \{j_0, j_1, \ldots, j_n\},\ \ell = 0, 1, \ldots, n\}.$$

If inserting city k^* at the position ℓ^* into the current tour yields a positive increase in the profit of the current tour, then we insert city k^* at the position ℓ^*. In this case, we have the tour $(j_0, j_1, \ldots, j_{\ell^*}, k^*, j_{\ell^*+1}, \ldots, j_n)$ with $n+1$ cities in it. Starting from the new tour with $n+1$ cities, we try to find another city to insert into the current tour until we cannot find a city providing a positive increase in the profit of the current tour.

The chart on the left side of the figure below shows 15 cities over a 10×10 geographical region. The distance associated with arc (i, j) is the Euclidean distance between cities i and j. The reward associated with visiting each city is indicated in brackets next to label of the city. For example, if we visit city 4, then we collect a reward of 2.1. We apply the greedy heuristic described above on the prize-collecting traveling salesman problem that takes place over these cities. The output of the greedy heuristic is shown on the right side of the figure below. The total profit from the tour is 23.06. The tour in the figure below may look reasonable, but we can improve this tour with simple inspection. For example, if we connect

city 4 to city 13, city 13 to city 0, city 0 to city 3 and city 3 to city 10, while keeping the rest of the tour unchanged, the profit from the new tour is 23.90. Note that this tour skips cities 12 and 14.



Construction heuristics are intuitive and they are not computationally intensive, but they often end up with solutions that are clearly suboptimal. In the figure above, since the portion of the tour that visits the cities 0, 3, 4, 12, 13 and 14 has a crossing and the distances of the arcs are given by the Euclidean distances between the cities, it was relatively simple to spot that we could improve this tour. In the next section, we discuss improvement heuristics that are substantially more powerful than construction heuristics.

3 Improvement Heuristics

In an improvement heuristic, we start with a certain solution. This solution could have been obtained by using a construction heuristic. We consider all solutions that are within the neighborhood of the current solution we have on hand. What we mean by a neighborhood of a solution depends on the specific problem we are working on and we shortly give examples within the context of the prize-collecting traveling salesman problem. Considering all solutions within the neighborhood of the current solution on hand, we pick the best solution within the neighborhood. If this best solution is not better than the current solution on hand, then we conclude that there are no better solutions within the neighborhood of the current solution and we stop. On the other hand, if this best solution is better than the current solution, then we update our current solution on hand to be this best solution. Starting from the new current solution on hand, we consider all solutions within the neighborhood of the new current solution on hand and the process repeats itself.

For the prize-collecting traveling salesman problem, we can define the neighborhood of a solution in many different ways. We may say that a solution $(i_0, i_1, \ldots, i_m)$ is in the neighborhood of the solution $(j_0, j_1, \ldots, j_n)$ if the solution $(i_0, i_1, \ldots, i_m)$ can be obtained by inserting one more city into the solution $(j_0, j_1, \ldots, j_n)$. That is, the neighborhood of the

solution $(j_0, j_1, \ldots, j_\ell, j_{\ell+1}, \ldots, j_n)$ is defined by all solutions of the form

$$(j_0, j_1, \ldots, j_\ell, k, j_{\ell+1}, \ldots, j_n)$$

for all choices of $k \in N \setminus \{j_0, j_1, \ldots, j_n\}$ and $\ell = 0, 1, \ldots, n$. For example, if the set of cities is $N = \{0, 1, 2, 3, 4, 5\}$ and the current solution we have on hand visits the cities $(0, 2, 3, 5)$, then the solutions in the neighborhood of this solution are

$$\begin{array}{ccccc} (0,1,2,3,5), & (0,2,1,3,5), & (0,2,3,1,5), & (0,2,3,5,1), & (0,4,2,3,5), \\ & (0,2,4,3,5), & (0,2,3,4,5), & (0,2,3,5,4). & \end{array}$$

Note that all of the solutions above are obtained by adding one more city into the current solution $(0, 2, 3, 5)$ we have on hand.

Similarly, we may say that a solution $(i_0, i_1, \ldots, i_m)$ is in the neighborhood of the solution $(j_0, j_1, \ldots, j_n)$ if the solution $(i_0, i_1, \ldots, i_m)$ can be obtained by removing one city from the solution $(j_0, j_1, \ldots, j_n)$. Therefore, the neighborhood of the solution $(j_0, j_1, \ldots, j_{\ell-1}, j_\ell, j_{\ell+1}, \ldots, j_n)$ is defined by all solutions of the form

$$(j_0, j_1, \ldots, j_{\ell-1}, j_{\ell+1}, \ldots, j_n)$$

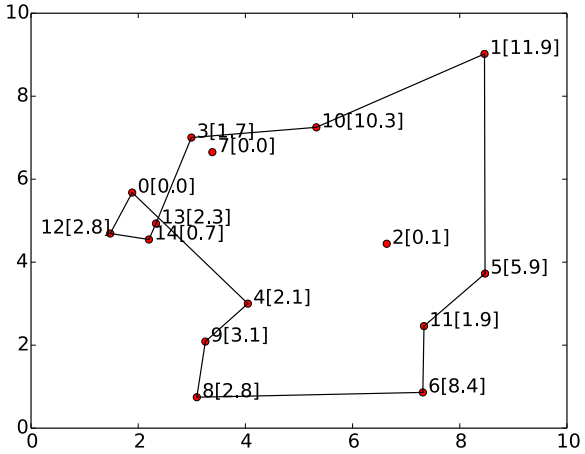
for all choices of $\ell = 1, 2, \ldots, n$. Following this definition of a neighborhood, if the set of cities is $N = \{0, 1, 2, 3, 4, 5\}$ and the current solution we have on hand visits the cities $(0, 2, 3, 5)$, then the solutions in the neighborhood of this solution are given by

$$(0,3,5), \quad (0,2,5), \quad (0,2,3).$$

We can join the two possible definitions of a neighborhood and say that a solution $(i_0, i_1, \ldots, i_m)$ is in the neighborhood of the solution $(j_0, j_1, \ldots, j_n)$ if the solution $(i_0, i_1, \ldots, i_m)$ can be obtained by either inserting one more city into or removing one city from the solution $(j_0, j_1, \ldots, j_n)$. In this case, if the set of cities is $N = \{0, 1, 2, 3, 4, 5\}$ and the current solution we have on hand visits the cities $(0, 2, 3, 5)$, then the solutions in the neighborhood of this solution are

$$\begin{array}{ccccc} (0,1,2,3,5), & (0,2,1,3,5), & (0,2,3,1,5), & (0,2,3,5,1), & (0,4,2,3,5), \\ (0,2,4,3,5), & (0,2,3,4,5), & (0,2,3,5,4), & (0,3,5), & (0,2,5), \quad (0,2,3). \end{array}$$

We check the performance of an improvement heuristic on the prize-collecting traveling salesman problem instance given in the previous section. First, we obtain an initial tour by using the greedy heuristic discussed in the previous section. Starting from this initial tour, we apply the improvement heuristic, assuming that a solution is in the neighborhood of the current solution if the solution can be obtained from the current solution by inserting a city into or removing a city from the current solution. The figure below shows the tour obtained by this improvement heuristic. The profit from this tour is 24.75. Recall that the profit from the tour obtained by the greedy heuristic alone was 23.06. The improvement heuristic provides about 7% more profit than the greedy heuristic alone.



4 More Elaborate Neighborhoods

More elaborate definitions of a neighborhood may allow us to search better and better solutions in our improvement heuristics. For example, for the prize-collecting traveling salesman problem, we may say that a solution $(i_0, i_1, \ldots, i_m)$ is in the neighborhood of the solution $(j_0, j_1, \ldots, j_n)$ if the solution $(i_0, i_1, \ldots, i_m)$ can be obtained by choosing a portion of the tour $(j_0, j_1, \ldots, j_n)$ and reversing the order of the cities in this portion. In other words, the neighborhood of the solution $(j_0, j_1, \ldots, j_k, j_{k+1}, \ldots, j_{\ell-1}, j_\ell, \ldots, j_n)$ is defined by all solutions of the form

$$(j_0, j_1, \ldots, j_\ell, j_{\ell-1}, \ldots, j_{k+1}, j_k, \ldots, j_n)$$

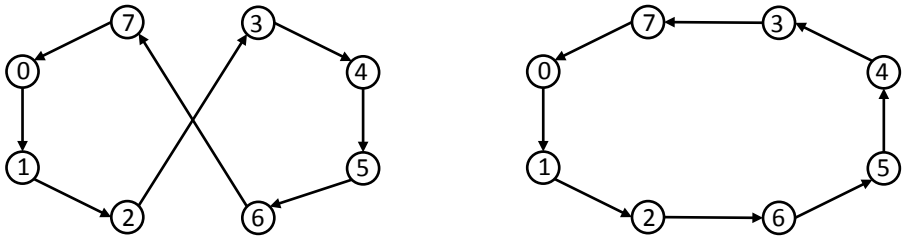
for all choices of $k, \ell = 1, 2, \ldots, n$ with $k < \ell$. For example, if the set of cities is $N = \{0, 1, 2, 3, 4, 5\}$ and the current solution we have on hand visits the cities $(0, 2, 3, 5)$, then the solutions in the neighborhood of this solution are given by

$$(0, 3, 2, 5), \quad (0, 5, 3, 2), \quad (0, 2, 5, 3).$$

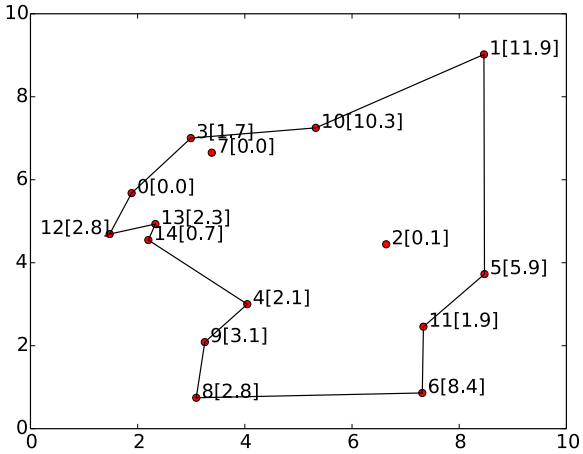
The first tour above is obtained by reversing the portion $(2, 3)$ in the tour $(0, 2, 3, 5)$, the second tour above is obtained by reversing the portion $(2, 3, 5)$ in the tour $(0, 2, 3, 5)$ and the third tour above is obtained by reversing the portion $(3, 5)$ of the tour $(0, 2, 3, 5)$.

In general, the definition of a neighborhood requires some insight into the problem on hand. For example, the definition of a neighborhood given above can be useful to remove crossings in a tour. To see how this, assume that the current solution we have on hand corresponds to the tour given on the left side of the figure below. The sequence of the cities visited in this tour is $(0, 1, 2, 3, 4, 5, 6, 7)$. This tour has a crossing. If we focus on the portion $(3, 4, 5, 6)$ of the tour and reverse the order of the cities visited in this portion, then we obtain the tour $(0, 1, 2, 6, 5, 4, 3, 7)$. We show this tour on the right side of the figure below. Note that the tour on the right side of the figure does not have the crossing on the left side. If the distances on the arcs are given by the Euclidean distances between the cities, then the length

of the tour on the right side must be shorter than the length of the tour on the left side. Since these two tours visit the same cities, they collect the same rewards. Thus, the profit from the tour on the right side is larger than the profit from the tour on the left side. In this example, we defined the neighborhood of a current solution on hand as all solutions that are obtained by reversing a certain portion of the tour in the current solution. In this case, if the current solution on hand has a crossing, then we can always find a solution in its neighborhood that provides better profit. Note that coming up with this neighborhood definition used some knowledge about the prize-collecting traveling salesman problem. In particular, we know that if the distances on the arcs are given by the Euclidean distances between the cities, then removing crossings in the tour improves the profit.



We check the performance of another improvement heuristic on the prize-collecting traveling salesman problem instance given earlier in this chapter. First, we obtain an initial tour by using the greedy heuristic. Starting from this initial tour, we apply the improvement heuristic, assuming that a solution is in the neighborhood of the current solution if the solution can be obtained from the current solution by inserting a city into or removing a city from the current solution or if the solution can be obtained by reversing a portion of the tour in the current solution. The figure below shows the tour obtained by this improvement heuristic. The profit of this tour is 26.08, which corresponds to 12% more profit than the greedy heuristic alone!



5 Final Remarks on Heuristics

How we define the neighborhood of a solution is a critical factor for the success of an improvement heuristic. In an improvement heuristic, we consider all solutions within the neighborhood of the current solution on hand. On one hand, the neighborhood of a solution should include a large number of other solutions because we want to consider a large number of possible solutions to improve the current solution on hand. On the other hand, we need to check the objective value provided by all solutions in the neighborhood. If the number of solutions in the neighborhood is astronomically large, then we cannot check the objective value provided by all solutions within the neighborhood of the solution we have on hand. Keeping the neighborhood of a solution rich enough is critical to find better solutions, but if we keep the neighborhood too rich, then checking the objective value of all solutions in the neighborhood gets time consuming.

In our discussion of improvement heuristics, we stated that an improvement heuristic checks all of the solutions in the neighborhood of the current solution on hand. If the best solution in the neighborhood is not better than the current solution on hand, then the improvement heuristic stops. Note that a good solution may not be in the neighborhood of the current solution on hand. Thus, an improvement heuristic has the risk stopping prematurely without obtaining a good solution. It is important to always remember that although heuristics tend to provide good solutions, they are not guaranteed to provide the optimal solution or even a good solution! There are more sophisticated improvement heuristics that that check all solutions within the neighborhood of the current solution on hand and update the current solution on hand to be the best solution in the neighborhood even if the best solution in the neighborhood is not better than the current solution on hand. The idea is that although we cannot find a better solution in the immediate neighborhood of the current solution on hand, there can be better solutions a few steps away in the neighborhood of the neighborhood of the current solution. Such improvement heuristics are generally known as simulated annealing and tabu search methods. They are precisely directed to address the possibility that a good solution may not be in the immediate neighborhood of the current solution on hand.

One of the frustrating shortcomings of heuristics is that they provide a solution, but we usually have no idea about how far this solution is from the optimal solution. This shortcoming is often overlooked in practice because if the solution provided by a heuristic is better than the status quo, then there is no reason not to implement the solution provided by the heuristic. Nevertheless, if we do not know how far the solution provided by a heuristic is from the optimal solution, then we can never be sure about when we should stop looking for a better solution or a more sophisticated heuristic. For this reason, proper optimization algorithms always have tremendous value. If we can formulate and solve a problem by using a proper optimization algorithm, then we should definitely choose that option over using heuristics. Sometimes, we can formulate a problem as an integer program, but we cannot

obtain the optimal solution in a reasonable amount of time. Even in those cases, we can solve the linear programming relaxation of the problem we formulated. The optimal objective value of the linear programming relaxation would be an upper bound on the optimal objective value of the problem we want to solve. In this case, we can try to compare the objective value provided by a heuristic solution with the upper bound on the optimal objective value of the problem. If the gap between the upper bound on the optimal objective value of the problem and the objective value from the heuristic is small, then we can safely conclude that the solution provided by the heuristic is near-optimal. Thus, even if we cannot solve the integer programming formulation of a problem exactly, linear programming relaxations of such formulations can provide useful information. Lastly, as mentioned at the beginning of this chapter, heuristic approaches can be used to complement the branch-and-bound method when solving an integer program. In particular, if we have a good solution provided by a heuristic, then we can use this solution to stop the search at many nodes of the tree during the course of the branch-and-bound method.

Optimization under Uncertainty

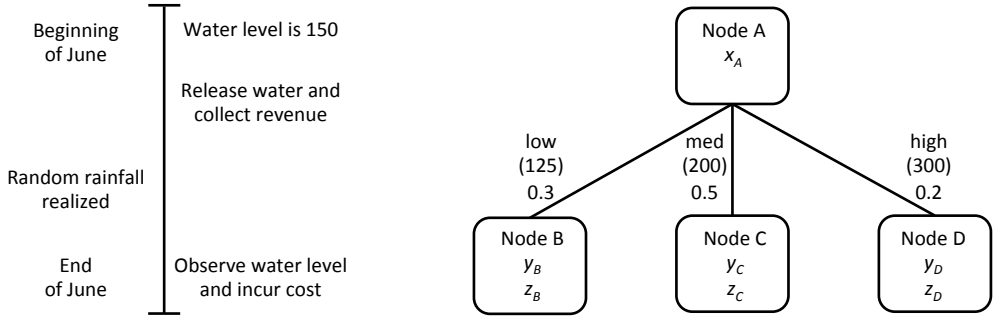
In some optimization problems, some parts of the data may be uncertain and we may need to make decisions now without knowing the future realization of the uncertain data. For example, we may need to decide how much inventory to purchase now without knowing what the demand for the product will be in the future or we may need to decide where to reposition the drivers now without knowing where the demand for the drivers will occur in the future. In this chapter, we discuss how we can solve optimization problems when there is uncertainty in some parts of the data and this uncertainty is revealed only later on.

1 Two-Stage Problems under Uncertainty

One class of optimization problems under uncertainty takes place over two stages. First, we make a set of decisions now and collect the reward associated with them. Then, we observe the outcome of a random quantity. After we observe the outcome of a random quantity, we make another set of decisions and collect the reward associated with these decisions. The goal is to maximize the total expected reward over the two stages. As an example, consider the following problem. We want to control the level of water in a reservoir over the single month on June. At the beginning of June, we have 150 units of water in the reservoir. At the beginning of June, we decide how much water to release from the reservoir. The water we release results in irrigation benefits and for each unit of water we release, we collect a revenue of \$3. After we decide how much water to release, we observe the random rainfall during the month. The rain fall can take three values, low, medium and high. Low rainfall occurs with probability of 0.3 and increases the water level in the reservoir by 125 units. Medium rainfall occurs with probability 0.5 and increases the water level by 200 units. High rainfall occurs with probability 0.2 and increases the water level by 300 units. Note that we must decide how much water we release from the reservoir before we see the realization of the random rainfall. At the end June, we observe the water level in the reservoir. The water in the reservoir has recreational benefits and we want to maintain a minimum water level of 100 units at the end of the month. If the water level at the end of June is below 100 units, when we incur a cost of \$5 for each unit short. The goal is to decide how much water to release at the beginning of June to maximize the total expected profit, where the total expected profit is given by the difference between the revenue from releasing water at the beginning of the month and the expected cost incurred when we are short of water at end of the month. The cost that we incur at the end of the month depends on the random rainfall. Therefore, the cost incurred at the end of the month is random as well. So, we are interested in the expected cost that we incur at the end of the month.

In this problem, we need to make decisions before and after the outcome of a random quantity becomes revealed to us. On the left side of the figure below, we show the time line of the events in the problem. At the beginning of June, we observe the water level in the reservoir, decide how much water to release and collect the revenue from the water that

we release. During the month, the random rainfall is realized. At the end of the month, we observe the water level, compute if and how much we are short of the desired water level and incur the cost associated with each unit of water we are short. On the right side of the figure, we give a tree that gives a more detailed description of the sequence of events and the decisions in the problem. The nodes of the tree correspond to the states of the world. The branches of the tree correspond to the realizations of the random quantities. Node A in the tree corresponds to the state of the world here and now at the beginning of June. The three branches leaving node A correspond to the three different realizations of the rainfall. Node B corresponds to state of the world at the end of June after having observed that the realization of the rainfall is low, at which point we need to check if and how much we are short of the desired water level and incur the cost for each unit of water we are short. The interpretations of nodes C and D are similar, but these nodes correspond to the cases where the rainfall was observed to be medium and high.



Next, we think about the decisions that we need to make at each node. As a result of this process, we will associate decision variables with each node in the tree. At node A, we decide how much water to release from the reservoir. Therefore, associated with node A, we define the following decision variable.

x_A = Amount of water released at node A.

At node B, we measure the level of water in the reservoir and we incur a cost for each unit we are short of the desired water level. Associated with node B, we define the following decision variables.

y_B = Given that we are at node B, water level in the reservoir.

z_B = Given that we are at node B, the amount we are short of the desired water level.

Note that y_B captures the water level at the end of June given that the rainfall was low during the month and z_B captures the amount we are short at the end of June given that the rain fall was low. We define the decision variables y_C, z_C, y_D and z_D with similar interpretations,

but these decision variables are associated with nodes C and D in the tree. All decision variables are indicated in the tree shown above. We proceed to constructing the objective function. For each unit of water we release at node A, we collect a revenue of \$3. Thus, revenue at node A is $3\,x_A$. At node B, we incur \$5 for each unit of water we are short. So, the cost at node B is $5\,z_B$. Also, the probability of reaching node B is 0.3, which is the probability of having low rainfall. Similarly, the costs incurred at nodes C and D are given by $5\,z_C$ and $5\,z_D$, whereas the probabilities of reaching these nodes are 0.5 and 0.2. So, we write the total expected profit obtained over the whole month of June as

$$3\,x_A - 0.3 \times 5\,z_B - 0.5 \times 5\,z_C - 0.2 \times 5\,z_D = 3\,x_A - 1.5\,z_B - 2.5\,z_C - z_D.$$

In the expression above, we multiply the cost incurred at each node by the probability of reaching that node to compute the total expected cost incurred at the end of June.

We now construct the constraints in the problem. The decision variable y_B corresponds to the water level at the end of June given that the rainfall during the month turned out to be low. The water level at the end of the month depends on how much water we had at the beginning of the month, how much water we released and the rainfall during the month. Noting that we have 150 units in the reservoir at the beginning of June, we release x_A units of water from the reservoir and low rainfall corresponds to a rainfall of 125 units, we can relate y_B to the decision variable x_A as

$$y_B = 150 - x_A + 125.$$

For each unit we are short of the desired water level of 100, we incur a cost of \$5. The decision variable z_B corresponds to the amount we are short given that the rainfall during the month turned out to be low. Thus, if y_B is less than 100, then $z_B = 100 - y_B$, whereas if y_B is greater than 100, then $z_B = 0$. To capture this relationship between the decision variables z_B and y_B, we use the constraints

$$z_B \geq 100 - y_B \qquad \text{and} \qquad z_B \geq 0.$$

Since z_B appears in the objective function with a negative coefficient and we maximize the objective function, we want to make the decision variable z_B as small as possible. If y_B is less than 100, then $100 - y_B \geq 0$. Therefore, due to the two constraints above, if y_B is less than 100, then the smallest value that z_B can take is $100 - y_B$. In other words, if y_B is less than 100, then the decision variable z_B takes the value $100 - y_B$, as desired. On the other hand, if y_B is greater than 100, then we have $100 - y_B \leq 0$. In this case, due to the two constraints above, if y_B is greater than 100, then the smallest value that z_B can take is 0. In other words. if y_B is greater than 100, then the decision variable z_B takes the value 0, as desired. By using the same argument, we have the constraints

$$\begin{aligned} &y_C = 150 - x_A + 200, \qquad z_C \geq 100 - y_C, \qquad z_C \geq 0, \\ &y_D = 150 - x_A + 300, \qquad z_D \geq 100 - y_D, \qquad z_D \geq 0. \end{aligned}$$

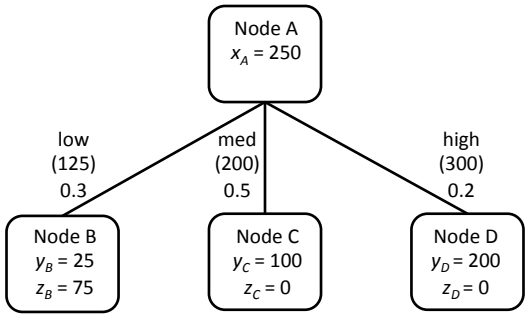
Therefore, to maximize the total expected profit over the month of June, we can solve the linear program

$$\begin{aligned}
\max \quad & 3\,x_A - 1.5\,z_B - 2.5\,z_C - z_D \\
\text{st} \quad & y_B = 150 - x_A + 125 \\
& z_B \geq 100 - y_B \\
& y_C = 150 - x_A + 200 \\
& z_C \geq 100 - y_C \\
& y_D = 150 - x_A + 300 \\
& z_D \geq 100 - y_D \\
& x_A, y_B, z_B, y_C, z_C, y_D, z_D \geq 0.
\end{aligned}$$

The optimal objective value of the problem above is 637.5 with the optimal values of the decision variables given by

$$x_A = 250, \quad y_B = 25, \quad z_B = 75, \quad y_C = 100, \quad z_C = 0, \quad y_D = 200, \quad z_D = 0.$$

We show the optimal solution in the tree below. According to the solution in the tree, we release 250 units of water at the beginning of June. If the rainfall turns out to be low, then the water level at the end of the month is 25 and we are 75 units short of the desired water level. If the rainfall turns out to be medium or high, then the water level at the end of the month is respectively 100 or 200, in which case, we are not short. Note that to maximize the expected profit, we release 250 units of water at the beginning of the month, which implies that we are willing to be short of the desired water level when the rainfall during the month turns out to be low. The revenue that we obtain from the released water justifies the cost incurred at the end of the month if the rainfall turns out to be low.



2 Multi-Stage Problems under Uncertainty

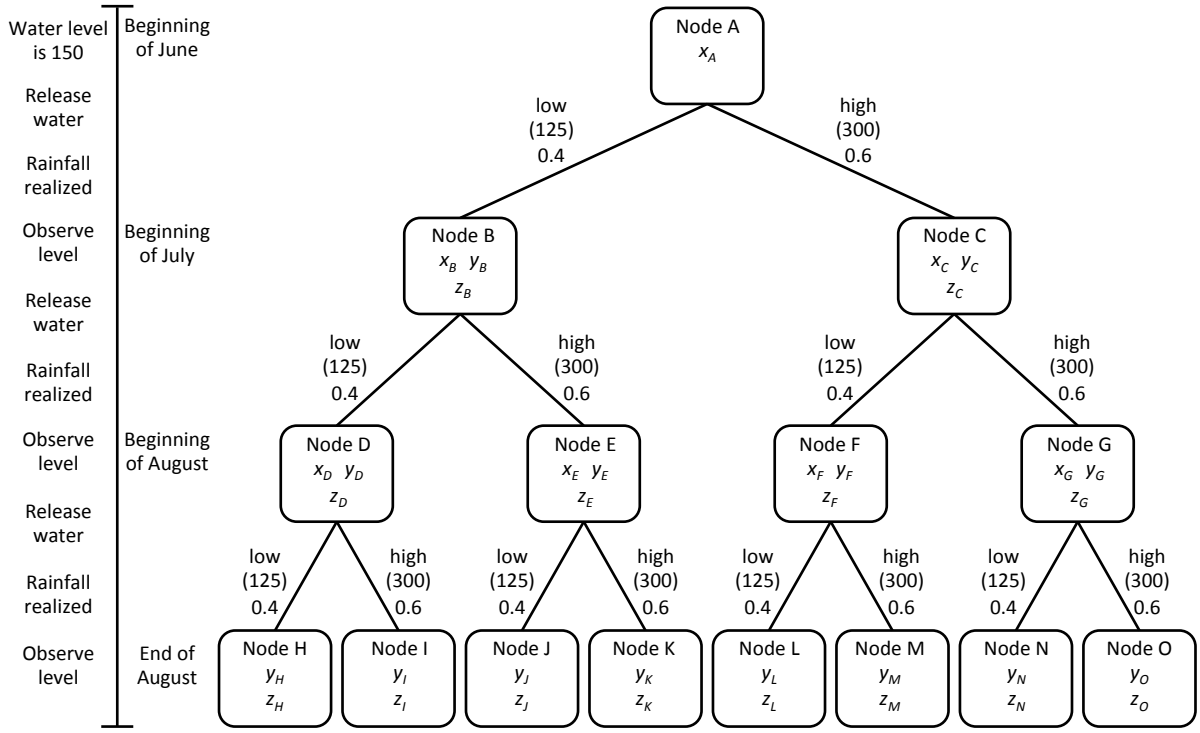
The problem we studied in the previous section takes place in two stages. We first make a set of decisions and collect the reward associated with these decisions. Then, we observe the outcome of a random quantity. After we observe the outcome of a random quantity, we

make another set of decisions and collect the reward associated with these decisions. For the problem in the previous section, the decisions in the second stage were rather simple. We simply measured the water level and calculated how much we are short of the desired level. In this section, we study problems that take place under uncertainty over multiple stages, instead of two stages. In multi-stage problems, we begin by making a set of decisions. Then, we observe the outcome of a random quantity. After we observe the outcome of a random quantity, we make another set of decisions. After making these decisions, we observe the outcome of another random quantity and make more decisions. The process of observing the outcome of a random quantity and making decisions continues until we reach the end of the planning horizon. For example, consider the problem of controlling the inventory of a certain product over 4 weeks. We first decide how much product to purchase. Then, we observe the random demand for the current week. After we observe the random demand, we decide how much to purchase in the next week. After making this decision, we observe the random demand in the next week. The process continues for 4 weeks. Our goal could be to maximize the expected profit given by the difference between the revenue from the demand we satisfy and the cost from purchasing the product.

We build on the example in the previous section to illustrate how we can model problems that takes place over multiple stages. Assume that we control the water level in a reservoir over 3 months, June, July and August. At the beginning of June, we have 150 units of water in the reservoir. At the beginning of each of the 3 months, we decide how much water to release from the reservoir. For each unit of water we release, we collect a revenue of \$3. During each month, we observe the random rainfall, which can take values low or high. Low rainfall occurs in each month with a probability of 0.4 and increases the water level in the reservoir by 125 units. High rainfall occurs with a probability of 0.6 and increases the water level in the reservoir by 300 units. At the end of each of the 3 months, we observe the water level. If the water level is below the desired level of 100 units, then we incur a cost of \$5 for each unit short. The goal is to decide how much water to release at the beginning of each of the 3 months to maximize the total expected profit over the 3 months.

On the left side of the figure below, we show the time line of the events. At the beginning of each month, we observe the water level in the reservoir and decide how much water to release. At the end of each month, we observe the water level and compute how much we are short of the desired water level. On the right side of the figure, we give a tree that gives a detailed description of the sequence of events and the decisions in the problem. Node A corresponds to the state of the world at the beginning of June. The two branches leaving node A correspond to the two possible realizations of the rainfall during June. For example, node B corresponds to the state of the world at the end of June and at the beginning of July, given that the rainfall during June turned out to be low. The nodes deeper in the tree represent the states of the world later in the planning horizon. For example, node F corresponds to the state of the world at the end of July and at the beginning of August, given that the rainfall during June was high and the rainfall during July was low. Similarly,

node J corresponds to the state of the world at the end of August, given that the rain fall during June, July and August was respectively low, high and low.



Let us think about the decision variables in the problem. At each node in the tree except for the leaf nodes that are at the very bottom, we need to decide how much water to release from the reservoir. Thus, we define the following decision variables.

$x_i =$ Given that we are at node i, the amount of water released from the reservoir, for $i = A, B, C, D, E, F, G$.

For example, the decision variable x_C represents how much water we release at the beginning of July given that we are at node C. In other words, x_C represents how much water we release at the beginning of July given that the rainfall during June was high. Similarly, x_E represents how much water we release at the beginning of August given that the rainfall during June and July was respectively low and high. Since the planning horizon ends at the end of August, we do not worry about how much water to release at the end of August. Thus, we do not worry about defining decision variables that capture the amount of water released at the nodes H, I, J, K, L, M, N and O. On the other hand, at each node in the tree except for the root node at the very top, we need to measure the level of water and how much we are short of the desired level. So, we define the following decision variables.

$y_i =$ Given that we are at node i, water level in the reservoir, for $i = B, C, D, E, F, G, H, I, J, K, L, M, N, O$.

$z_i =$ Given that we are at node i, the amount we are short of the desired water level, for $i = B, C, D, E, F, G, H, I, J, K, L, M, N, O$.

The water level in the reservoir at node A is known to be 150. Therefore, we do not need decision variables that measure the water level and how much we are short of the desired water level at node A.

Next, we construct the objective function in the problem. Each node in the tree contributes to the expected profit. As an example, we consider node G. At node G, the amount of water we release is given by the decision variable x_G. Thus, we make a revenue of $3\,x_G$. At this node, the amount we are short of the desired water level is given by the decision variable z_G. Thus, we incur a cost of $5\,z_G$ at node G. So, the profit at node G is given by $3\,x_G - 5\,z_G$. We reach node G when the rainfall in June and July are respectively high and high. Thus, the probability of reaching node G is $0.6 \times 0.6 = 0.36$. In this case, the contribution of node G to the expected profit is given by $0.36\,(3\,x_G - 5\,z_G)$. Considering all the nodes in the tree, the objective function is given by

$$\begin{aligned}
&3\,x_A + 0.4\,(3\,x_B - 5\,z_B) + 0.6\,(3\,x_C - 5\,z_C) \\
&\quad + 0.16\,(3\,x_D - 5\,z_D) + 0.24\,(3\,x_E - 5\,z_E) + 0.24\,(3\,x_F - 5\,z_F) + 0.36\,(3\,x_G - 5\,z_G) \\
&\quad - 0.064 \times 5\,z_H - 0.096 \times 5\,z_I - 0.096 \times 5\,z_J - 0.144 \times 5\,z_K - 0.096 \times 5 z_L \\
&\quad - 0.144 \times 5\,z_M - 0.144 \times 5\,z_N - 0.216 \times 5\,z_O.
\end{aligned}$$

We proceed to constructing the constraints in the problem. For each node in the tree, we need to construct a constraint that computes the water level at the current node as a function of the water level at the parent node of the current node, the amount of water released at the parent node and the rainfall over the branch that connects the current node to its parent node. For example, for nodes B, G and H, we have the constraints

$$y_B = 150 - x_A + 125, \qquad y_G = y_C - x_C + 300, \qquad y_H = y_D - x_D + 125.$$

Furthermore, for each node in the tree, we need to compute how much we are short of the desired water level. For example, for nodes B, G and H, we compute how much we are short of the desired water level by using the constraints

$$z_B \geq 100 - y_B, \quad z_B \geq 0, \quad z_G \geq 100 - y_G, \quad z_G \geq 0, \quad z_H \geq 100 - y_H, \quad z_H \geq 0.$$

The idea behind the constraints above is identical to the one we used when we formulated the two-stage problem in the previous section. We construct the two types of constraints for all nodes in the tree except for node A. Since the water level at node A is known to be 150 units, we do not need to compute the water level and how much we are short at node A. Putting the discussion in this section together, we can maximize the total expected

profit over the 3 month planning horizon by solving the linear program

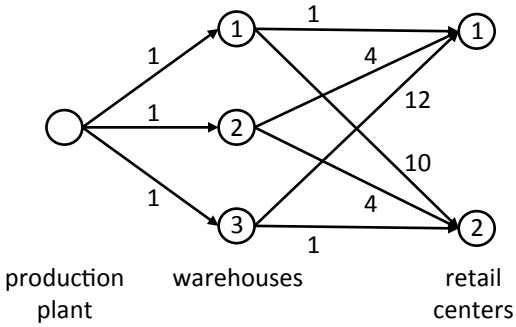
$$
\begin{aligned}
\max \quad & 3\,x_A + 0.4\,(3\,x_B - 5\,z_B) + 0.6\,(3\,x_C - 5\,z_C) \\
& + 0.16\,(3\,x_D - 5\,z_D) + 0.24\,(3\,x_E - 5\,z_E) + 0.24\,(3\,x_F - 5\,z_F) + 0.36\,(3\,x_G - 5\,z_G) \\
& - 0.064 \times 5\,z_H - 0.096 \times 5\,z_I - 0.096 \times 5\,z_J - 0.144 \times 5\,z_K - 0.096 \times 5 z_L \\
& - 0.144 \times 5\,z_M - 0.144 \times 5\,z_N - 0.216 \times 5\,z_O \\
\text{st} \quad & y_B = 150 - x_A + 125 \\
& z_B \geq 100 - y_B \\
& y_C = 150 - x_A + 300 \\
& z_C \geq 100 - y_C \\
& y_D = y_B - x_B + 125 \\
& z_D \geq 100 - y_D \\
& \vdots \\
& y_G = y_C - x_C + 300 \\
& z_G \geq 100 - y_G \\
& y_H = y_D - x_D + 125 \\
& z_H \geq 100 - y_H \\
& \vdots \\
& y_O = y_G - x_G + 300 \\
& z_O \geq 100 - y_O \\
& x_A, x_B, y_B, z_B, \ldots, x_G, y_G, z_G, y_H, z_H, \ldots, y_O, z_O \geq 0.
\end{aligned}
$$

The optimal objective value of the problem above is 2005. There are quite a few decision variables in the problem. Thus, we go over the optimal values of only a few of the decision variables. For example, we have $x_E = 700$ and $y_E = 575$ in the optimal solution. According to this solution, given that the rainfall during June and July was respectively low and high, it is optimal to release $x_E = 700$ units of water at the beginning of August. Given that the rainfall during June and July was respectively low and high, the optimal water level at the beginning of August is $y_E = 575$ units.

3 A Larger Two-Stage Problem under Uncertainty

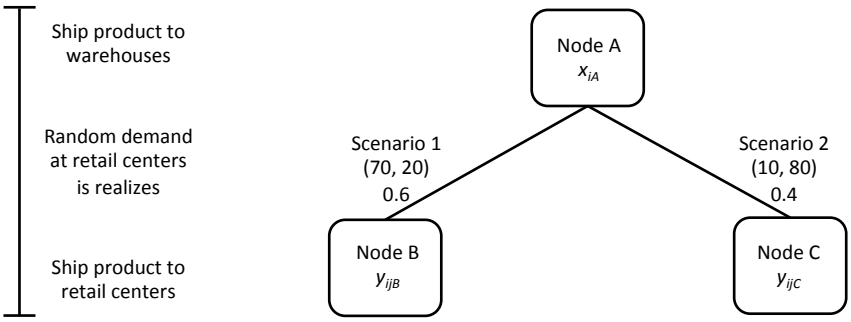
In some applications, the decisions that we make at each node of the tree may be captured by a large number of decision variables. To give an example, consider the situation faced by a company shipping products from its production plant to the warehouses and from the warehouses to its retail centers with the purpose of satisfying the random demand at the retail centers. At the beginning of the planning horizon, the company has 100 units of product at the production plant. It needs to decide how much to ship from the production plant

to each one of the 3 warehouses. After the company ships products to the warehouses, the random demand at the retail centers is realized. Once the company observes the realization of the random demand at the retail centers, it needs to decide how much product to ship from the warehouses to the retail centers to cover the demand. On the left side of the figure below, we depict the production plant, warehouses and retail centers. The label on each arc shows the cost of shipping a unit of product over each arc. For example, it costs \$1 to ship one unit from the production plant to each one of the warehouses and \$4 to ship one unit from warehouse 2 to retail center 1. On the right side of the figure, we show the possible demand realizations. In particular, there are two possible scenarios for demand realizations. The first scenario happens with probability 0.6. Under this scenario, the demand at retail centers 1 and 2 are respectively 70 and 20. The second scenario happens with probability 0.4. Under this scenario, the demand at retail centers 1 and 2 are respectively 10 and 80. Note that under scenario 1, the demand at retail center 1 is high, whereas under scenario 2, the demand at retail center 2 is high. So, it is hard to judge where the large demand will occur. The goal of the company is to minimize the total expected cost of satisfying the demand, where the cost includes the cost of shipping products from the production plant to the warehouses and from the warehouses to the retail centers.



Scenario	Prob.	Dem. at Ret. Cen. 1	Dem. at Ret. Cen. 2
1	0.6	70	20
2	0.4	10	80

On the left side of the figure below, we show the time line of the events. At the beginning, we decide how much product to ship to each warehouse. Then, we observe the realization of the demands. After observing the realization of the demands, we decide how much product to ship from the warehouses to the retail centers to cover the demands. On the right side of the figure, we give a tree that shows a more detailed description of the sequence of events and the decisions in the problem. Node A in the tree corresponds to the state of the world here and now. At this node, we decide how much product to ship to the warehouses. The two branches leaving node A correspond to the two demand scenarios given in the table above. Node B corresponds to the state of the world where the demands turned out to be the one in scenario 1. At this node, we need to decide how much product to ship from the warehouses to the retailer centers. Similarly, node C corresponds to the state of the world where the demands turned out to be the one in scenario 2. At this node, we also need to decide how much product to ship from the warehouses to the retailer centers. To capture the decisions in the problem, we define the following decision variables.



$x_{iA} =$ Given that we are at node A, amount of product shipped to warehouse i, for $i = 1, 2, 3$.

$y_{ijB} =$ Given that we are at node B, amount of product shipped from warehouse i to retail center j, for $i = 1, 2, 3,\ j = 1, 2$.

$y_{ijC} =$ Given that we are at node C, amount of product shipped from warehouse i to retail center j, for $i = 1, 2, 3,\ j = 1, 2$.

We indicate these decision variables in the tree shown above. Note that since node B corresponds to the case where the demands turned out to be the one in scenario 1, the decision variables $\{y_{ijB} : i = 1, 2, 3,\ j = 1, 2\}$ capture the products shipped from the warehouses to the retail centers under scenario 1. Since it costs \$1 to ship one unit of product from the production plant to each one of the warehouses, the cost incurred at node A is $\sum_{i=1}^{3} x_{iA}$. For notational brevity, we use c_{ij} to denote the cost of shipping a unit of product from warehouse i to retail center j. So, the costs incurred at nodes B and C are $\sum_{i=1}^{3}\sum_{j=1}^{2} c_{ij}\, y_{ijB}$ and $\sum_{i=1}^{3}\sum_{j=1}^{2} c_{ij}\, y_{ijC}$. Since the probabilities of reaching nodes B and C are 0.6 and 0.4, the total expected cost can be written as

$$\sum_{i=1}^{3} x_{iA} + 0.6 \sum_{i=1}^{3}\sum_{j=1}^{2} c_{ij}\, y_{ijB} + 0.4 \sum_{i=1}^{3}\sum_{j=1}^{2} c_{ij}\, y_{ijC}.$$

Next, we construct the constraints in the problem. At node A, the total amount of product that we ship out of the production plant cannot exceed the product availability at the production plant. Therefore, we have the constraint

$$\sum_{i=1}^{3} x_{iA} \leq 100.$$

At node B, the total amount of product that we ship out of each warehouse i cannot exceed the amount of product shipped to the warehouse. Noting that the amount of product shipped to warehouse i is given by x_{iA}, for all $i = 1, 2, 3$, we have the constraint

$$\sum_{j=1}^{2} y_{ijB} \leq x_{iA}.$$

Furthermore, at node B, the amount of product shipped to each retail center should be enough to cover the demand at the retail center. At node B, the demands at the retail centers are observed to be 70 and 20. Thus, we have the constraints

$$\sum_{i=1}^{3} y_{i1B} \geq 70 \qquad \text{and} \qquad \sum_{i=1}^{3} y_{i2B} \geq 20.$$

In this case, to figure out how to ship the products from the production plant to the warehouses and from the warehouses to the retail centers to minimize the total expected cost, we can solve the linear program

$$\begin{aligned}
\max \quad & \sum_{i=1}^{3} x_{iA} + 0.6 \sum_{i=1}^{3} \sum_{j=1}^{2} c_{ij}\, y_{ijB} + 0.4 \sum_{i=1}^{3} \sum_{j=1}^{2} c_{ij}\, y_{ijC} \\
\text{st} \quad & \sum_{i=1}^{3} x_{iA} \leq 100 \\
& \sum_{j=1}^{2} y_{ijB} \leq x_{iA} \qquad \forall\, i = 1, 2, 3 \\
& \sum_{i=1}^{3} y_{i1B} \geq 70 \\
& \sum_{i=1}^{3} y_{i2B} \geq 20 \\
& \sum_{j=1}^{2} y_{ijC} \leq x_{iA} \qquad \forall\, i = 1, 2, 3 \\
& \sum_{i=1}^{3} y_{i1C} \geq 10 \\
& \sum_{i=1}^{3} y_{i2C} \geq 80 \\
& x_{iA}, y_{ijB}, y_{ijC} \geq 0 \qquad \forall\, i = 1, 2, 3,\ j = 1, 2.
\end{aligned}$$

The optimal objective value of the problem above is 340. We focus on the values of some of the decision variables in the optimal solution. In particular, the optimal solution has $x_{1A} = 20$, $x_{2A} = 50$ and $x_{3A} = 30$. There are two interesting observations. First, observe that the costs of shipping products from warehouse 1 to retail center 2 and from warehouse 3 to retail center 1 are rather high. Thus, if we ship a large amount of product to warehouse 1 and retail center 2 ends up having a large demand, then we incur a high cost to cover the demand at retail center 2. Similarly, if we ship a large amount of product to warehouse 3 and retail center 1 ends up having a large demand, then we incur a high cost to cover the

demand at retail center 1. In contrast, the costs of shipping products from warehouse 2 to either of the retail centers is moderate. In the optimal solution, we ship a relatively large amount of product to warehouse 2. After observing the realization of the demand at the retail centers, we use the products at warehouse 2 to satisfy the demand. Second, although the total amount of demand at the retail centers never exceeds 90, the total amount of product that we ship to the warehouses is 100. The idea is that there is value in having products at warehouses 1 and 3. If the demand at retail center 1 turns out to be large, then we can use the products at warehouse 1, rather than the products at warehouse 2 or 3. Similarly, if the demand at retail center 2 turns out to be large, then we can use the products at warehouse 3, rather than the products at warehouse 1 or 2. Since the cost of shipping the product from the production plant to the warehouses is quite low, in this problem instance, it turns out that it is optimal to ship all of the products out of the production plant to keep the warehouses stocked.