MS&E 111/ENGR 62: Supplementary Notes

Ashish Goel

Autumn 2007-08, Stanford University
(The notes will continually evolve; last modified September 29, 2008)

Optimization permeates many (almost all) branches of human endeavor. Darwinian evolution optimizes a species for the "best traits" for survival. Firms optimize their supply chains. Fund managers optimize their portfolios. On a lighter note, personal dating sites try to find the "best match", and advertisers on online sites try to show you the "best advertisement" based on your user profile. The window arch, the suspension bridge, the coding scheme used by your cell phone, are all highly optimized. This class will largely study Linear Programming (or LPs), one of the simplest and perhaps the most widely used paradigms for optimization. We will see lots of real world problems that can be modeled and solved as Linear Programs. We will study and use properties of LPs such as duality and vertex optimality. We will spend significant amount of time on a special sub-class of LPs, known as network flow problems. Finally, we will use LPs as a base to briefly explore two other important techniques in optimization: Dynamic Programming and Convex Optimization.

1 The knapsack problem

In this lecture, we will use the knapsack problem as an example to give you a preview of several basic concepts from linear programming. Consider a thief who goes into a warehouse with a knapsack. There is a limit to how much weight the knapsack can carry. The warehouse has many items, each of which is divisible into arbitrarily small fragments eg. gold, silver, diamond dust, etc. The thief wants to fill her knapsack with the most profitable bundle of goods. How can she decide what this bundle is?

1.1 Modeling

The first stage in solving an optimization problem is modeling. And the very first as well as the most important step in modeling is to *name your quantities*[1]. Let the knapsack capacity be W lbs. Let N denote the number of goods in the warehouse. Let the total weight of the i-th good be w_i and let the value of the i-th good be v_i.

The next step is to decide what quantities we are free to choose. These are known as **decision variables**. In this case, the thief must decide how much of each item to carry. Let x_i denote the fraction of the i-th good carried away by the thief.

The next two steps in modeling an optimization problem are to decide what our goals and restrictions are. The goal is commonly referred to as an **objective function** which we need to either maximize or minimize depending on the problem. In this case, the objective function is $\sum_{i=1}^{N} v_i x_i$, which is the total profit made by the thief, and we need to maximize the objective function. The restrictions are that (a) the thief can not take more of a good than is available in the warehouse, and (b) the thief can not carry more weight than the knapsack capacity. Using our notation, we have:

$$\begin{aligned} (a)\ \forall i \in \{1, \ldots, N\}, \quad x_i &\leq 1 \quad \text{and} \\ (b)\ \sum_{i=1}^{N} x_i w_i &\leq W. \end{aligned}$$

When we write down the restrictions mathematically as above, we call them **constraints**. The symbol $\forall$ is the "for all" symbol, and the symbol $\in$ is the "belongs to" symbol. Thus the first constraint (a) is really a set of N constraints, one for each i in the set $\{1, \ldots, N\}$. Some other notation that we will commonly use is

$\sum$	:	Sum
$\prod$	:	Product
$\Re$	:	The set of real numbers
$\mathcal{Z}$	:	The set of integers
$\mathcal{Z}^+ . \Re^+$	:	The set of *non-negative* integers/real numbers respectively.
s.t.	:	subject to
A^T	:	The transpose of the matrix A.

[1]If you get stuck on a problem, try thinking about that problem without using any pronouns.

We will introduce more notation as we go along. But for now, let us return to the knapsack problem. Do we have all the restrictions in place? As it turns out we missed one very important restriction. The thief can not steal negative quantities of any good. So we also need to add the constraints $(c) \forall i \in \{1, \ldots, N\}, x_i \geq 0$. We will commonly write these constraints as a single constraint $x \geq 0$. This is *vector notation* and we are using x to denote the vector $\langle x_1, x_2, \ldots x_N \rangle$; more on that later.

We now have our complete model. The problem, in mathematical terms is:

$$\begin{array}{ll} \text{maximize} & \displaystyle\sum_{i=1}^{N} x_i v_i \\ \text{subject to:} & \\ & \begin{array}{rrcl} \forall i \in \{1, \ldots, N\}: & x_i & \leq & 1 \\ & \sum_{i=1}^{N} x_i w_i & \leq & W \\ & x & \geq & 0. \end{array} \end{array}$$

This is an example of what is known as a "linear program" i.e. an optimization problem where the objective function is linear in the decision variables and each constraint is an inequality involving only linear functions of the decision variables. The more general term "mathematical program" is used to describe an optimization problem where the objective function and the constraints are more general.

1.2 Solving the problem

The next stage after modeling is to solve the problem. Of course we need to specify the values of all the parameters we used (i.e. N, W, w_i, v_i etc.) in the modeling part. This is known as an **instance** of the problem. Consider the problem where we have three goods (say gold, diamond, and silver) and the problem parameters are $N = 3, W = 4, w = \langle 2, 3, 4 \rangle, v = \langle 5, 20, 3 \rangle$. Here we have again used vector notation to succinctly write down the values of v_1, v_2, v_3 and w_1, w_2, w_3 as single vectors v an w. The above model now reduces to:

$$\begin{array}{ll} \text{maximize} & 5x_1 + 20x_2 + 3x_3 \\ \text{subject to:} & \end{array}$$

$$\begin{array}{rrrcl} x_1 & & & \leq & 1 \\ & x_2 & & \leq & 1 \\ & & x_3 & \leq & 1 \\ 2x_1 + & 3x_2 + & 4x_3 & \leq & 4 \\ & & x & \geq & 0. \end{array}$$

We will solve this problem using Excel. The solution that we obtain is $x_1 = 0.5, x_2 = 1, x_3 = 0$. This is what we call an **optimum solution**. The maximum profit that the thief can obtain is 22.5 . We call this the **optimum value** of the objective function. Often, we will refer to both the optimum value and the optimum solution together as the "optimum". In this instance, there is a unique optimum solution, but that may not always be the case. The solution $x_1 = 1, x_2 = 0, x_3 = 0.5$ also satisfies all the constraints. In general, any solution (i.e. assignment of values to decision variables) that satisfies all the constraints is a **feasible solution**.

Linear programs can be solved efficiently. Excel works well for small linear programs. Other commercial packages such as CPLEX and free packages such as lpsolve are better suited to larger LPs. These solvers use multiple algorithms, including the classic simplex method pioneered by George Dantzig, and more recent approaches such as interior point methods. In this class, we will not dig deep into these algorithms.

1.3 Analysis and interpretation

Having solved the problem, are we done? Not at all – this is where the fun starts. Linear programs satisfy a range of properties which often provide striking and useful insight into the original problem. We will provide a sneak preview of two such properties. We will treat them in more detail later.

1. Only one of the goods (gold) in the above example is chosen fractionally. Diamonds are chosen completely, whereas silver is entirely left behind. In general, for a knapsack problem with N goods, only one good will be chosen fractionally by solvers such as Excel. A full proof of this statement will follow later in the class. But here is the intuition for this specific instance. The optimum solution for this instance of the knapsack problem is unique. To uniquely define three unknowns, we need three linear equations. Thus at least three of our inequalities must be satisfied with equality, i.e. be **tight**. Where can these three inequalities come from? One of these can be (and in this case, is) the knapsack capacity constraint $2x_1 + 3x_2 + 4x_3 \leq 4$. But that still leaves

2 more. Clearly, $x_1 \leq 1$ and $x_1 \geq 0$ can not both be tight. The same for x_2 and x_3. Hence, any good can contribute only one tight constraint. Thus, the two tight constraints must come from two different goods, in which case neither of those two goods can be chosen fractionally. This leaves only one good which can be chosen fractionally in the optimum solution.

2. Suppose a crime syndicate wants to buy out the thief. They offer to pay the thief a price y_1 for the gold, a price y_2 for the diamonds, a price y_3 for the silver, and a price y_4 per lb for the knapsack. But the thief can use 2 lbs of knapsack capacity and all her gold to generate a profit of 5 units, so $2y_4 + y_1$ should be at least 5. Similarly, $3y_4 + y_2 \geq 20$ and $4y_4 + y_3 \geq 3$. The syndicate would like to minimize the total price it pays, i.e. minimize $y_1 + y_2 + y_3 + 4y_4$. Also, the prices should be non-negative, otherwise the thief will hold on to that resource (gold, diamond, silver, or knapsack capacity). This gives us the following LP which the syndicate can use to decide the minimum price:

$$\begin{array}{rlllll}
\text{minimize} & y_1 + y_2 + y_3 + 4y_4 & & & & \\
\text{subject to:} & & & & & \\
 & y_1 + & & & 2y_4 & \geq 5 \\
 & & y_2 + & & 3y_4 & \geq 20 \\
 & & & y_3 + & 4y_4 & \geq 3 \\
 & & & & y & \geq 0.
\end{array}$$

Try solving this linear program in Excel. It turns out that $y = \langle 0, 12.5, 0, 2.5 \rangle$ is an optimum solution, giving an optimum value of 22.5, or, the same as the optimum value of the previous LP.

In retrospect, the connection is not really surprising. What is the minimum total price the thief would settle for? Clearly 22.5, since this is how much the thief can make by filling her knapsack and walking away from any deal being offered by the syndicate. This is an example of duality. The variables y_1, y_2, y_3, y_4 are called dual variables or dual prices. Linear programming duality will be a central theme in this class. It is a very important concept in economics; in fact prices in the real world can often best be understood as dual variables.

1.4 Discussion points

1. The above instance had a unique optimum solution. Will this be true for all instances of the knapsack problem?
2. What happens if we remove the constraint $x \geq 0$?
3. Can you list some more feasible solutions to this LP?
4. Can you think of more realistic applications for the knapsack problem?
5. Why might it be practically important that only one good is chosen fractionally?

As the number of variables increases, the original approach used to solve this problem in Excel becomes tedious. Remember, we had to individually label all the variables, and do some computation for each constraint. Vector notation is going to be very useful in expressing and solving LPs, and that will be our next topic.

2 The knapsack problem in vector notation

First, refamiliarize yourself with basic matrix multiplication and dot products. Next, consider the knapsack LP that we wrote above but let us add some more terms so that each inequality explicitly involves all the variables:

$$\begin{array}{ll} \text{maximize} & 5x_1 + 20x_2 + 3x_3 \\ \text{subject to:} & \\ & \begin{array}{rrrcl} x_1 + & 0x_2 + & 0x_3 & \leq & 1 \\ 0x_1 + & x_2 + & 0x_3 & \leq & 1 \\ 0x_1 + & 0x_2 + & x_3 & \leq & 1 \\ 2x_1 + & 3x_2 + & 4x_3 & \leq & 4 \\ & & x & \geq & 0. \end{array} \end{array}$$

This can succinctly be written as

$$\begin{array}{ll} \text{maximize} & c^T \cdot x \\ \text{subject to:} & \\ & \begin{array}{rcl} Ax & \leq & b \\ x & \geq & 0. \end{array} \end{array}$$

where c is the column vector $\begin{pmatrix} 5 \\ 20 \\ 3 \end{pmatrix}$, A is the matrix $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 2 & 3 & 4 \end{pmatrix}$, b is the column vector $\begin{pmatrix} 1 \\ 1 \\ 1 \\ 4 \end{pmatrix}$, and x is the column vector of decision variables $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$. We can then set up this problem in Excel without having to assign names to individual variables and having to write down individual constraints. For this small instance, this might not seem like much saving of work. But imagine an instance with 100s of goods. We can just type in and name the individual matrices (often the matrices are already entered into a spreadsheet since the data has to come from somewhere), use MMULT and SUMPRODUCT to express the constraints, and we are done. More details will follow in the Excel solver tutorial.

The representation

$$\begin{array}{rl} \text{maximize} & c^T \cdot x \\ \text{subject to:} & \\ & \begin{array}{rcl} Ax & \leq & b \\ x & \geq & 0. \end{array} \end{array}$$

is one of the standard ways of writing a linear program. This can be used to naturally represent the optimization problem VRM[2] 1.2.1, for example. If a problem has n variables and m constraints, then c and x are $n \times 1$ vectors, b is an $m \times 1$ vector, and A is an $m \times n$ matrix. Verify that all the matrix dimensions are consistent.

There are other standard ways of writing linear programs, but they are all equivalent (i.e. a LP in one standard form can be transformed into a LP in another standard form). For example, one general form for a LP is:

$$\begin{array}{rl} \text{maximize/minimize} & c^T \cdot x \\ \text{subject to:} & \\ & \begin{array}{rcl} Ax & \leq & b \\ Dx & \geq & f \\ Gx & = & h. \end{array} \end{array}$$

[2]VRM refers to the notes by Van Roy and Mason on the class web page

where c, x are $n \times 1$ vectors, and A, D, G are $m_1 \times n$,$m_2 \times n$,$m_3 \times n$ matrices, respectively. If the objective is to minimize $c^T \cdot x$ we can define $r = -c$ and the objective now becomes to maximize $r^T \cdot x$; else we define $r = c$. We can define $S = -D$ and $t = -f$ and the constraint $Dx \geq f$ becomes $Sx \leq t$. We can further define $U = -G$ and $v = -h$ and the constraint $Gx = h$ can be replaced by the two constraints $Gx \leq h$ and $Ux \leq v$. We can now use W to represent the matrix $\begin{pmatrix} A \\ -D \\ G \\ -G \end{pmatrix}$ and z to represent the vector $\begin{pmatrix} b \\ -f \\ h \\ -h \end{pmatrix}$. The LP now becomes

$$\begin{array}{rl} \text{maximize} & r^T \cdot x \\ \text{subject to:} & \\ & Wx \;\leq\; z. \end{array}$$

The matrix W is $(m_1+m_2+2m_3)\times n$ dimensional. This is almost in standard form except that we are missing the constraint $x \geq 0$. The next trick is to replace each decision variable by a difference of two decision variables. So $x = p - q$. Even if we constrain p, q to be non-negative, x is not constrained. So the above LP can now be written as

$$\begin{array}{rl} \text{maximize} & r^T \cdot p - r^T \cdot q \\ \text{subject to:} & \\ & \begin{array}{lcl} Wp - Wq & \leq & z \\ p, q & \geq & 0. \end{array} \end{array}$$

Finally, (having exhausted almost all the letters of the alphabet!) we can define $J = (W \quad -W)$, $k = \begin{pmatrix} r \\ -r \end{pmatrix}$ and the decision variables $y = \begin{pmatrix} p \\ q \end{pmatrix}$ we get our standard form

$$\begin{array}{rl} \text{maximize} & k^T \cdot y \\ \text{subject to:} & \\ & \begin{array}{lcl} Jy & \leq & z \\ y & \geq & 0. \end{array} \end{array}$$

This is more than a technical exercise. We now know that any theorems we prove for our standard theorem will apply to general LPs. Another commonly used standard form is:

$$\begin{array}{rlcl} \text{minimize} & b^T \cdot y & & \\ \text{subject to:} & & & \\ & Ay & \geq & c \\ & y & \geq & 0. \end{array}$$

3 An example: matching men and women

Example 3.1 *Consider the problem where we are given 3 men, Dave, Eddie, and Frank, and 3 women, Gloria, Helen, and Iris. We would like to pair the 3 men and 3 women, such that there is no polygamy and total compatibility is maximized. Let C be a 3×3 matrix where $C_{i,j}$ is the compatibility of the i-th man with the j-th woman. Let us define decisions variables $x_{i.j}$ to indicate whether the i-th man is married to the j-th woman. This problem can be modeled as*

$$\begin{array}{lll} \textit{maximize} & \displaystyle\sum_{i=1}^{3}\sum_{j=1}^{3} C_{i,j}x_{i,j} & \\ \textit{subject to:} & & \\ & \forall i \in \{1,2,3\} \quad \sum_{j=1}^{3} x_{i,j} & = \quad 1 \\ & \forall j \in \{1,2,3\} \quad \sum_{i=1}^{3} x_{i,j} & = \quad 1 \\ & \qquad\qquad\qquad x & \geq \quad 0. \end{array}$$

Reduce this to either of the two standard forms. Solve this problem (in any form, not necessarily a standard form) using Excel for the following compatibility matrix:

Compatibility	*Gloria*	*Helen*	*Iris*
Dave	*1*	*0*	*0.5*
Eddie	*0.75*	*2.0*	*1.0*
Frank	*0.5*	*2.5*	*1.5*

A little observation shows that this instance has at least two optimum solutions. Is there a fractional optimum solution? Does Excel return a fractional or an integer solution? Mentally file your observations away for future reference.

It is clear how to extend this problem to N men and N women. Observe that the task assignment problem (VRM 1.0.1) is the same mathematical problem even though it arises in a different context. In this class, we will often discuss seemingly toy problems such as knapsack, matching men and women, etc. These problems capture a wide range of real life applications. For example, knapsack and knapsack like problems are used to maximize profit constrained by resources, as in VRM 1.2.1.

4 A useful linear programming modeling technique

Often, linear programming can be used to model objective functions and constraints that may not seem linear at first glance. We will see two such examples: Minimizing the max and maximizing the min. We will start with an example, then state the general technique, and give some more examples.

Example 4.1 Minimize Makespan: *We have N jobs and M agents. Agent i can complete job j in time $t_{i,j}$. A job can be split among two agents, and an agent can perform multiple jobs. The completion time of the last job to be completed is known as the "makespan" of the problem. Our goal is to minimize the makespan.*

Solution: Assume that our decision variables are $x_{i,j}$, which represent the fraction of job j assigned to agent i, and y_i which represent the total time required for agent i to complete all the jobs assigned to agent i. Then, we have the constraints:

$$\begin{array}{rrcl} \forall j \in \{1, \dots, N\}: & \sum_{i=1}^{M} x_{i,j} & = & 1 \\ \forall i \in \{1, \dots, M\}: & \sum_{j=1}^{N} x_{i,j} t_{i,j} & = & y_i \\ & x, y & \geq & 0. \end{array}$$

But what would the objective function be? Here, we will indulge in a bit of "bait and switch": we will look at the problem in a different light. Our original goal is to minimize the time when all the jobs get completed. But this is the same as the time when all the agents are done. So our new goal is to minimize the largest of the y_i's, i.e., our objective function is

$$\text{minimize } \max_{i=1}^{M} y_i.$$

The function $\max_{i=1}^{M} y_i$ is not a linear function. How then can we capture this as a linear program? The answer is simple. We introduce a new variable z and add a new set of M constraints

$$\forall i \in \{1, \dots, M\} : z - y_i \geq 0.$$

We can then simply use the objective function

$$\text{minimize } z$$

to obtain a linear program. ■

In the above example, since for any feasible solution, $z \geq y_i$ for all i, it follows that we must also have $z \geq \max_i y_i$. But since the objective function attempts to minimize z and z does not appear in any other constraints, *at the optimum point* we must have $z = \max_i y_i$. In general, we can use this technique to minimize the maximum of any number of linear functions, or maximize the minimum of any number of linear functions.

Exercise 4.2 *Is it necessary to introduce $z \geq 0$ as a constraint in the above linear program? What will be the impact of introducing this constraint?*

Suppose we have a set of linear constraints with n variables $x_1, x_2, \dots, x_n$, i.e. the decision variables form an $(n \times 1)$ vector. Suppose we also have k $(n \times 1)$ vectors $c_1, c_2, \dots, c_k$. Then,

1. The objective function "minimize $\max_{i=1}^{k} c_i^T x$" can be modeled as a linear program by introducing a new decision variable z, k new constraints "$\forall i \in \{1, \dots, k\} : z \geq c_i^T x$", and by changing the objective function to "minimize z".
2. The objective function "maximize $\min_{i=1}^{k} c_i^T x$" can be modeled as a linear program by introducing a new decision variable z, k new constraints "$\forall i \in \{1, \dots, k\} : z \leq c_i^T x$", and by changing the objective function to "maximize z".

Example 4.3 *Minimizing the absolute value: The absolute value of a quantity α is merely the maximum of α and $-\alpha$. Hence, the absolute value can be minimized using the same technique as above.*

Example 4.4 *Min/max/absolute value constraints: We can impose the constraint "$\min_{i=1}^{k} c_i^T x \geq \alpha$" by using the k constraints*

$$\forall i \in \{1, \dots, k\} : c_i^T x \geq \alpha.$$

We can impose the constraint "$\max_{i=1}^{k} c_i^T x \leq \alpha$" by using the k constraints

$$\forall i \in \{1, \dots, k\} : c_i^T x \leq \alpha.$$

We can impose the constraint "$|c^T x| \leq \alpha$" by writing $|c^T x|$ as $\max\{c^T x, -c^T x\}$ and then using the technique for max.

Unfortunately, we can not use the above technique to represent the constraints "$\min_{i=1}^{k} c_i^T x \leq \alpha$" or "$\max_{i=1}^{k} c_i^T x \geq \alpha$" or "$|c^T x| \geq \alpha$".

Exercise 4.5 *Consider the functions* $a(x) = x_1 + 2x_2$ *and* $b(x) = 2x_1 + x_2$. *Graph the regions defined by each of the following:* $\min\{a(x), b(x)\} \geq 3$, $\min\{a(x), b(x)\} \leq 3$, $\max\{a(x), b(x)\} \leq 3$, $\max\{a(x), b(x)\} \geq 3$, $|a(x) - b(x)| \geq 1$ *and* $|a(x) - b(x)| \leq 1$. *Intuitively, which of these regions looks like a polytope (i.e. an intersection of half-spaces)? Identify the half-spaces.*

5 Ten steps towards understanding basic feasible solutions

In this section, we will assume that U denotes the set of feasible solutions of a given LP, i.e. the set of all points in $\Re^N$ which satisfy all the constraints. Recall that U is an intersection of half-spaces, and that we were using the term "polytope" to refer to such a set. We will assume that the linear program under consideration has N variables and M constraints.

1. Convex sets and convex combinations Given two points x and y in N dimensional space, and any real number α between 0 and 1 (inclusive, i.e. $0 \leq \alpha \leq 1$ or more concisely, $\alpha \in [0, 1]$), the point $z = \alpha x + (1 - \alpha)y$ is called a convex combination of x and y. The set of all convex combinations of x and y is the line segment joining x and y.

Given K points $b_1, b_2, \ldots, b_K$ in $\Re^N$, and given K non-negative real numbers $\alpha_1, \alpha_2, \ldots, \alpha_K$ such that $\sum_{i=1}^{K} \alpha_i = 1$, the point $z = \sum_{i=1}^{K} \alpha_i b_i$ is called a convex combination of $b_1, b_2, \ldots, b_K$. A convex combination can also be thought of as a "weighted average". The set of all convex combinations of 3 points is the filled triangle with these points as vertices[3].

Exercise 5.1 *Given points x, y, z, suppose a is a convex combination of x, y and b is a convex combination of a, z. Show that b is also a convex combination of x, y, z.*

A set S of points from N dimensional space is said to be convex if for all $x, y \in S$, all convex combinations of x, y are also in S.

Some examples of convex sets are cubes, cuboids, spheres, cones, line segments, infinite lines, etc. Please do not confuse convex sets with convex/concave functions.

[3]The set of all convex combinations of $b_1, b_2, \ldots, b_K$ is also known as the convex hull of $b_1, b_2, \ldots, b_K$.

2. Half spaces are convex Consider the half space defined by $a^T x \le d$ where a is a given $N \times 1$ vector, x is an $N \times 1$ variable vector, and d is a scalar (i.e., a 1×1 matrix). This half space consists of all points which lie on or one side of the hyperplane defined by $a^T x = d$. Suppose x, y are two points in this half-space. Let z be a convex combination of x, y, i.e., $z = \alpha x + (1 - \alpha) y$ for some $\alpha \in [0, 1]$. Since x, y are in the half-space, we must have

$$a^T x \le d, \quad \text{and}$$

$$a^T y \le d.$$

Multiplying the first inequality by α and the second inequality by $1 - \alpha$, and adding, we get

$$a^T(\alpha x + (1 - \alpha) y) \le d$$

which implies $a^T z \le d$, i.e., z is also in the half-space. Hence the half-space is convex.

The reason that we could multiply the two original inequalities by α, $1 - \alpha$ respectively without changing signs is that $\alpha \in [0, 1]$. Always be careful while multiplying inequalities; if you multiply with a negative number, the inequality flips.

3. Intersection of two convex sets is convex Suppose S, T are two convex sets, x, y are two points in $S \cap T$, and z is a convex combination of x, y. Since x, y are in $S \cap T$ they are in S. Since S is convex, this implies that z is also in S. Similarly, z is also in T, and hence, in $S \cap T$. This proves that $S \cap T$ is convex.

4. The set of feasible solutions to a linear program is convex Since the feasible solution to a linear program is an intersection of half-spaces, it must be convex by steps 2,3 above. In other words, all polytopes are convex.

5. Basic feasible solution Recall that U denotes the set of feasible solutions to a given linear program. A point $x \in \Re^N$ is said to be a basic feasible solution if

1. $x \in U$, i.e., x is feasible, and
2. There do not exist two other (i.e., different from x) feasible points y, z such that x is a convex combination of y, z.

Informally, basic feasible solutions are "extreme points", i.e., they can not be represented as convex combinations of other feasible points. They are also known as "vertex" solutions and "corner-point solutions".

Let $b_1, b_2, \ldots, b_K$ refer to the basic feasible solutions of the linear program under consideration. K can be 0, i.e., it is possible for a linear program to not have any basic feasible solution.

6. Basic feasible solutions and bounded polytopes If U is bounded, then any point in U is a convex combination of the basic feasible solutions, $\mathrm{b}_1, b_2, \ldots, b_K$.

We will omit the proof of this statement; you can see VRM 3.2.4 for a proof.

7. Basic feasible solutions and optimality for bounded polytopes

Theorem 5.2 (Equivalent to VRM 3.3.1) *If U is bounded, then there exists an optimum solution that is a basic feasible solution.*

Note that the theorem does not claim that all optimum solutions are basic feasible; just that there exists one.

Proof: Assume, without loss of generality, that the objective function is to maximize $c^T x$. Let x be an optimum solution. From step 6, we conclude that x is a convex combination of $b_1, b_2, \ldots, b_K$, i.e., there exist non-negative real numbers $\alpha_1, \alpha_2, \ldots, \alpha_K$ such that $\sum_{i=1}^K \alpha_i = 1$ and $x = \sum_{i=1}^K \alpha_i b_i$. If $c^T x > c^T b_i$ for all $i \in \{1, \ldots, K\}$ then we must have $\sum_{i=1}^K \alpha_i c^T x > \sum_{i=1}^K \alpha_i c^T b_i$ which is impossible since the right hand side is $c^T \sum_{i=1}^K \alpha_i b_i$ which is equal to $c^T x$ and the left hand side is equal to $c^T x \sum_{i=1}^K \alpha_i$ which is also equal to $c^T x$. Hence, there must exist at least one $i \in \{1, \ldots, K\}$ such that $c^T x \leq c^T b_i$. Since x is an optimum solution, we can not have any other feasible solution yield a larger value for the objective function. Hence, b_i is also an optimum solution, and we have demonstrated the existence of an optimum solution that is basic feasible. ■

An equivalent statement of the above theorem is that if U is bounded, and x is an optimum solution among all basic feasible solutions, then x is an optimum solution among all points in U.

8. Basic feasible solutions and optimality for general polytopes We will state the following theorem (same as VRM 3.3.2) without proof.

Theorem 5.3 *If a linear program has an optimum solution as well as a basic feasible solution, then there exists an optimum solution which is also basic feasible.*

Thus, whenever the notion of "optimum" and "basic feasible" makes sense for an LP, we just have to look among basic feasible solutions to find an optimum.

Example 5.4 *The LP, maximize $3x + 4y$ subject to $3x + 4y \leq 7$ has an optimum solution but no basic feasible solution.*

Example 5.5 *The LP maximize $x + y$ subject to $x \geq 0, y \geq 0$ has a basic feasible solution but no optimum solution, i.e., the optimum is unbounded.*

9. Two implications of optimality of basic feasible solutions The first implication is for how LPs are solved. The famous simplex algorithm, developed by the famous George Dantzig, "walks" from basic feasible solution to basic feasible solution, improving the objective function as it goes along. By steps 8 and 9, this is sufficient to solve most LPs (perhaps it is fair to say, all interesting LPs). Hence, optimality of basic feasible solutions has had a tremendous impact on our ability to solve large LPs, even when computers were not very powerful. More than 50 years after it was developed, the simplex algorithm and its variants remain the most commonly used method for solving large LPs. In fact, all commercial LP solvers that we know of will return a basic feasible solution by default (unless you muck around with configuration parameters etc, eg. uncheck the "assume linear model" button in Excel)[4]. We will not discuss solution procedures for LPs in any more detail in this class.

The second implication is about the properties of LPs. For many types of LPs, basic feasible solutions have very interesting properties. Since LP solvers return optimum basic feasible solutions by default, we can safely assume that the optimum solutions that we find also have those interesting properties. This is a simple but surprisingly powerful tool; we can use

[4]Other algorithms may find optimum solutions in the interior, but commercial solvers which use those algorithms employ "cross-over" methods which translate these solutions to a basic feasible solution. To understand this (very) informally, imagine that an algorithm finds an optimum solution in the interior of a bounded polytope. Then, there must be a "degree of freedom" i.e. a direction in which we can perturb the optimum solution so that it remains both feasible and optimum. We can move in this direction till we hit the hyperplane defined by another constraint, and our "degrees of freedom" get reduced by 1. Repeating this process will put us at a basic feasible solution which is also optimum.

this to show that the knapsack problem can have at most one fractional solution, and that the maximum compatibility matching problem will not yield solutions that result in "fractional marriages". We have had a foretaste of these results earlier, and will soon prove them formally.

10. A useful property of basic feasible solutions One general property of basic feasible solutions is particularly useful. At least N constraints must be *binding* i.e. satisfied with equality (also sometimes called "tight") at a basic feasible solution. This is not surprising since we need N equalities to define a point in N dimensional space. We will omit a formal proof.

In fact, something stronger is true: N *linearly independent* constraints must be binding at a basic feasible solution; a set of inequalities is said to be linearly independent if the left hand side (i.e. the side with all the variables) of any inequality in the set can not be derived as a linear combination of the left hand sides of other inequalities in that set.

6 The knapsack problem revisited

6.1 Basic feasible solutions

Consider the polytope defined by the knapsack constraints:

$$\begin{array}{rccc} \forall i \in \{1,\ldots,N\}: & x_i & \leq & 1 \\ & \sum_{i=1}^{N} x_i w_i & \leq & W \\ \forall i \in \{1,\ldots,N\}: & x_i & \geq & 0. \end{array}$$

Just the first set of constraints above and the non-negativity constraints imply that the feasible region is contained in an N-dimensional "cube" of side 1. Hence, the feasible region for this linear program is bounded. This is an important step towards applying theorem VRM 3.3.1 to claim that there exists an optimal solution that is basic feasible. Please do not ignore this step[5].

Hence, we know that there exists at least one optimum solution that is basic feasible; let this solution be x. At the point x, at least N constraints must be binding. One of them can be the constraint $\sum_{i=1}^{N} x_i w_i \leq W$, but that still means that (at least) $N-1$ other constraints of the form $x_i \leq 1$ or $x_i \geq 0$ must be binding. For any good i, at most one of the constraints

[5]Of course you can apply the more general theorem VRM 3.3.2, for which you need to show both that there exists at least one basic feasible solution and at least one optimum solution.

$x_i \geq 0$ and $x_i \leq 1$ can be binding. Hence, the remaining (at least) $N-1$ binding constraints must correspond to (at least) $N-1$ different goods. Hence, (at least) $N-1$ goods satisfy one out of $x_i \geq 0$ and $x_i \leq 1$ and are therefore not chosen fractionally; the (at most) one remaining good can be chosen fractionally.

7 Minimum cost flow

"Minimum cost flow" is a problem formulation that can be used to model a wide range of real life problems. It can also be solved as a linear program, and its basic feasible solutions (and its dual, which we see later), have many interesting properties.

We will start with a warm up example, followed by a formulation of the general min cost flow problem and the properties of its basic feasible solutions. We will then use this general formulation to capture several interesting real life problems.

7.1 Warm-up example: a simple shortest path problem

Consider the graph in figure 1. Imagine that each node (i.e. small filled dot) in this graph represents a "city", and each edge in this graph represents a "road". We will commonly use V to denote the set of nodes and E to denote the set of edges in a graph. We will commonly refer to the graph as $G = (V, E)$. The arrow on a road represents the direction in which you can travel, and the number on the road represents the cost of using the road. Depending on the application, the cost might be the toll you have to pay, the amount of gas you will consume, or the time you will spend. Our goal is to find the minimum cost path from city s to city t. This is an instance of the shortest path problem. We will commonly use s to denote the "source node" and t to denote the "terminal node" for shortest path and related problems.

Please remember that the use of V, E, s, t, c as described above is a matter of convention (not a hard and fast rule); we reserve the right to use other notation where it makes more sense in an application. We will commonly use N to denote the number of vertices and M the number of edges in a graph.

How do we determine the shortest path using an LP? For any edge (v, w), i.e., for a road going from city v to city w, let $c_{v,w}$ denote its cost, and let $x_{v,w}$ denote a decision variable that captures whether this road is used in the shortest path or not. If $x_{v,w} = 0$ we will interpret that as the road not being

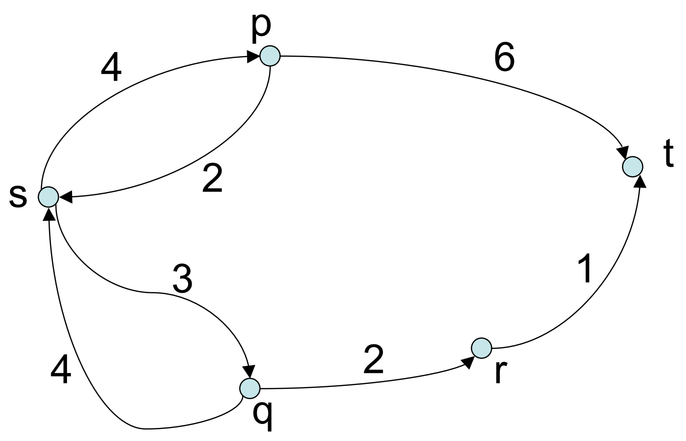

Figure 1: A shortest path example.

used at all, and if $x_{v,w} = 1$ we will interpret that as the road being used completely. If $x_{v,w}$ lies between 0 and 1, we will interpret that as "fractional use" of the road.

Our objective function is clearly to minimize the total cost of used roads, which is simply to minimize $\sum_{(v,w)\in E} c_{v,w}x_{v,w}$. We also need $x \geq 0$. What other constraints do we need? First, if we enter node p, we must also leave it, since p is neither the source nor the terminal. Thus the total amount of "entering" we do into p must be the same as the total amount of "leaving" from p, or, the net amount of leaving must be 0. The same is true for nodes r and q. These are commonly called conservation constraints, and can be mathematically expressed as:

$$\begin{array}{llcl} x_{p,s} + x_{p,t} & -x_{s,p} & = & 0 \\ x_{q,r} + x_{q,s} & -x_{s,q} & = & 0 \\ x_{r,t} & -x_{q,r} & = & 0. \end{array}$$

But the net amount of leaving from s must be 1 and from t must be -1. These are also called conservation constraints, and can be written as:

$$\begin{array}{llcl} x_{s,p} + x_{s,q} & -x_{p,s} - x_{q,s} & = & 1 \\ & -x_{r,t} - x_{p,t} & = & -1. \end{array}$$

This completes our LP. As it turns out, when you solve it, you will find $x_{v,w} = 1$ or 0 for every edge (v, w). This is because all basic feasible solutions of the above LP are simple paths (i.e. without cycles) from s to t. This holds in general, and we will explore it in the context of the more general min cost flow problem.

7.2 Formulation of min cost flow

In the min cost flow problem, we need to ship some goods from a set of nodes in a graph (the supply nodes) to another set of nodes (the demand nodes). We will capture that uniformly by using d_v to denote the demand of node v; if v is a supply node, d_v will be negative. Not all supply nodes necessarily have the same d_v, and not all demand nodes necessarily have the same d_v. In this problem, every demand must be exactly met, and every supply must be exactly exhausted.

We will refer to the amount of goods shipped on an edge as the "flow" on that edge. The intuition comes from thinking about edges as pipelines and the goods as a fluid such as water or oil. Each edge has a capacity, and can carry no more flow than its capacity. Each edge also has a cost-per-unit-flow. Our goal is to send flow from the supply nodes to the demand nodes in a minimum cost fashion. while respecting the capacity constraints and satisfying all the supplies and demands. Let $u_{v,w}$ represent the capacity of edge $(v_, w)$ and $c_{v,w}$ its cost. Again, while u, d, c are terms we will try to use consistently to denote capacities, demands, and costs, this is just a matter of convention, and we reserve the right to use different notation.

We will have similar decision variables as for the shortest path problem. Let $x_{v,w}$ denote the amount of flow on edge (v, w) . The objective function is simple, as before:

$$\text{minimize} \sum_{(v,w)\in E} c_{v,w} x_{v,w}.$$

The so called *capacity constraints* are simple as well:

$$\forall (v, w) \in E : \qquad x_{v,w} \le u_{v,w},$$

and of course we need $x \ge 0$. Observe that the total amount of flow leaving a supply node minus the total amount of flow entering that node must be equal to the amount of supply at the node. Similarly, the total amount of flow entering a demand node minus the total amount of flow leaving that node must be equal to the amount of demand at the node. These can be captured concisely[6] as: *the net amount of flow leaving a node must be equal to the negative of the demand.*

[6] Remember that we are using negative demand to indicate supply.

For node v, this can be written mathematically as:

$$\sum_{w:(v,w)\in E} x_{v,w} - \sum_{w:(w,v)\in E} x_{w,v} = -d_v.$$

The above constraint is known as the “flow conservation constraint”. adding this constraint for all the nodes completes the min cost flow LP:

$$\text{minimize} \sum_{(v,w)\in E} c_{v,w}x_{v,w}$$

subject to:

$$\begin{array}{lrcl}
 & \textit{CAPACITY CONSTRAINTS} & & \\
\forall (v,w) \in E: & x_{v,w} & \leq & u_{v,w} \\
 & & & \\
 & \textit{CONSERVATION CONSTRAINTS} & & \\
\forall v \in V: & \sum_{w:(v,w)\in E} x_{v,w} - \sum_{w:(w,v)\in E} x_{w,v} & = & -d_v \\
 & & & \\
 & x & \geq & 0.
\end{array}$$

We will use the notation $\text{OUT}_x(v)$ to denote $\sum_{w:(v,w)\in E} x_{v,w}$ and $\text{IN}_x(v)$ to denote $\sum_{w:(w,v)\in E} x_{w,v}$, and hence, the conservation constraint for node v can be written as $\text{OUT}_x(v)-\ \text{IN}_x(v) = -d_v$, in short.

Exercise 7.1 *How will you model the case where the total demand is less than the total supply, and where any unused supply can be discarded at no extra cost?*

Exercise 7.2 *How will you model the case where the total demand is less than the total supply, and where any unused supply can be discarded at an extra cost of α per unit?*

Example 7.3 *The shortest path problem can be modeled as a minimum cost flow problem by setting the demand (in the minimum cost flow problem) for the source node s to be -1, the demand for the terminal node t to be $+1$, the demands for all other nodes to 0, the costs in the minimum cost flow problem to be the same as the those in the shortest path problem and setting all the capacities to either 1 or ∞. This is an example of a* reduction*: the shortest path problem has been reduced to the minimum cost flow problem, and hence theorems that we prove about minimum cost flows will translate to shortest paths.*

All the flow conservation constraints can be written concisely as $Ax = -d$ where the matrix A has N rows (one for each node in the graph) and M columns (one for each edge in the graph). The entry in the row corresponding to node p and edge (v, w) is 1 if $p = v$, is -1 if $p = w$, and is 0 otherwise. This matrix is known as the *Incidence matrix*, and can be computed using a formula in Excel if the list of edges is given.

The Excel spreadsheets shortestold.xls and shortest.xls illustrate how to set up the min-cost flow problem using the example in figure 1. The first of these solves the problem exactly as specified, with capacities 1. The second sheet solves the same problem but the model ignores the capacity constraints (i.e., sets capacities to ∞) and also illustrates how the matrix A can be generated automatically.

Example 7.4 *The maximum compatibility matching problem (section 3) can also be modeled as a minimum cost flow problem by making a graph with N men and N women. We draw an edge from each woman w to each man m with cost equal to the negative of the compatibility, and capacity either 1 or* ∞. *Set the demand of all women to be* -1 *and that for all men to be* $+1$.

7.3 Basic feasible solutions of min cost flows

The following theorem is very useful:

Theorem 7.5 *If a minimum cost flow problem has integer demands and capacities, then any basic feasible solution must have integer flow on each edge.*

The proof is a simple cycle canceling technique. Details are in the proof of VRM 5.2.1 and rather than repeating the details, we will present an example of this technique here. Consider the shortest path example we have already seen but with a slight modification as described in figure 2: the cost of the edge (p, t) has been reduced to 3 so that the top path (s, p, t) and the bottom path (s, q, r, t) are equally expensive. Now consider the solution which sends flow 0.5 on the top path and 0.5 on the bottom (figure 3). This is an optimum solution (and in fact if you uncheck the "assume linear model" option, this is the solution you will get in the current version of Excel.).

Pick any edge with fractional flow in this solution, say the edge (s, p). The first endpoint of this edge is s. Write the node s and the edge (s, p) in a table.

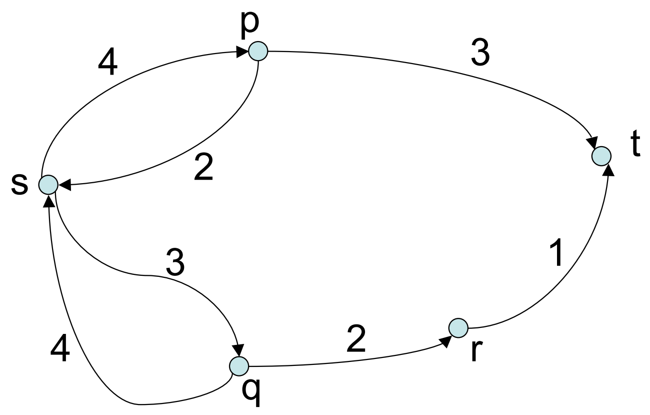

Figure 2: The shortest path example modified to have two possible shortest paths.

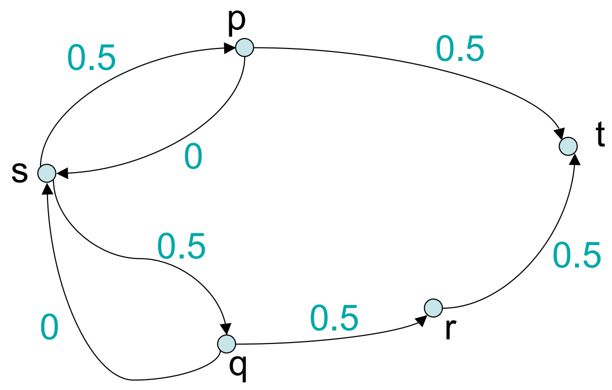

Figure 3: A fractional optimum solution for the modified shortest path example.The numbers on the edges are flows.

Nodes	Edges
s	(s,p)

Consider the conservation constraint for node p, the other endpoint of edge (s,p):

$$x_{p,s} + x_{p,t} - x_{s,p} = 0.$$

The flow on $x_{s,p}$ is fractional but the RHS of the above equation is integral (in this case, 0), and hence there must be another edge with fractional flow incident on p (i.e. with p as an endpoint). Indeed, there is such an edge, in particular, edge (p,t). Add p and (p,t) to the table:

Nodes	Edges
s	(s,p)
p	(p,t)

Consider the conservation constraint for t, the other endpoint of (p,t):

$$-x_{r,t} - x_{p,t} = -1.$$

Again, we already know that edge (p,t) has fractional flow. The RHS is again integral, and hence there must be another edge incident on t which has fractional flow. Indeed, such an edge exists, and is (r,t). We will add t and (r,t) to our table:

Nodes	Edges
s	(s,p)
p	(p,t)
t	(r,t)

Consider the conservation constraint for r, the other endpoint of (r,t):

$$x_{r,t} - x_{q,r} = 0.$$

Again, the RHS is integral, and we have already identified $x_{r,t}$ as fractional. Hence, another edge incident on r must have fractional flow. In this case, (q,r) is such an edge. We add r and (q,r) to the table:

Nodes	Edges
s	(s,p)
p	(p,t)
t	(r,t)
r	(q,r)

Consider the other endpoint of (q, r), i.e. q. The conservation constraint for q is:

$$x_{q,r} + x_{q,s} - x_{s,q} = 0.$$

Again, the RHS is integral, and we have already identified (q, r) as an edge that has fractional flow. Hence, another edge incident on q must have fractional flow. In this case, (s, q) is such an edge; we add this edge to our table along with node q:

Nodes	Edges
s	(s, p)
p	(p, t)
t	(r, t)
r	(q, r)
q	(s, q)

There is of course a point to this tedious repetition, and it is time to get to the punch line. The other endpoint of (s, q) is s, which is already in the table. *We have cycled back.* Whenever the demands are integral, we will be able to cycle in this fashion, *which is the first step in the general proof.*

Our cycle is (s, p, t, r, q, s). Notice that some of the edges we identified in the table go along this cycle, and some go against. But none of the edges in the table is at capacity (since the flows on these edges are all fractional) and none of the edges has 0 flow (again, since all the flows are fractional). Hence, for each of these edges, there is some small amount by which the flows can be increased and decreased freely without violating the capacity or non-negativity constraints. *This is the second step in the general proof.* Observe that the first step crucially used the fact that demands are integral, while the second crucially uses the fact that capacities are integral. In this example, the amount by which the flow on any edge in the table can be increased or decreased freely is 0.5, but this amount may be different for other examples. Choose any amount smaller than this number; in this case let us choose 0.1.

Let us send additional flow of 0.1 along this cycle (i.e. increase flow by 0.1 on the edges in the table that go along this cycle and decrease flow by 0.1 on edges that go against.) This gives the solution in figure 4(a). This must also satisfy the conservation constraints since the IN and OUT flows of each node on the cycle are increased by the same amount. Remember that we already ensured that the capacity and non-negativity constraints are satisfied as well, and hence this new solution is also feasible. Now send a flow of 0.1 against the cycle to obtain the feasible solution in figure 4(b).

The average of these two new feasible solutions gives us the original fraction solution from figure 3 and hence the solution in figure 3 can not be basic feasible. *This is the third step in the general proof.*

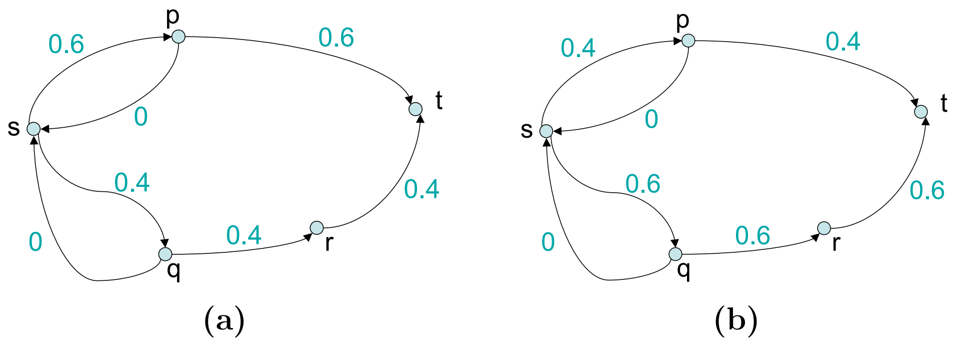

Figure 4: (a) Sending flow along the cycle (s, p, t, r, q, s). (b) Sending flow against the cycle (s, p, t, r, q, s).

We can always follow these three steps to prove theorem 7.5 in general. This theorem also has interesting corollaries about shortest paths and matchings, since we already showed that these problems can be reduced to minimum cost flow.

Corollary 7.6 *Any basic feasible solution to the maximum compatibility matching problem must be integral.*

Corollary 7.7 *Any basic feasible solution to the shortest path problem must be integral.*

In fact, basic feasible solutions to the shortest path problem have a very nice structure if the capacities u are set to ∞: any simple path (i.e. no cycles) from s to t is a basic feasible solution, and these are the only basic feasible solutions. With capacities set to ∞, if there is a cycle of edges of negative cost, then there is no optimum solution (since sending infinite flow around that cycle results in cost $-\infty$); in all other cases, if there is a path from s to t, commercial solvers such as Excel will return a simple path.

If the goal is to find a negative cost cycle in a graph (eg. for some arbitrage purposes) then set all demands to 0, all capacities to 1, and solve the minimum cost flow problem. If the optimum solution value is 0, then there is no negative cost cycle. If the optimum solution value is -ve, there is a negative cost cycle and the solution returned by the LP can be used to deduce such a cycle (the cycle will be composed of edges with $x_{v,w} = 1$).

Exercise 7.8 *Suppose you are given a variant of the minimum cost flow problem where there is a lower bound $l_{v,w}$ on the flow on edge (v, w) as well as an upper bound $u_{v,w}$, i.e. we have constraints $x_{v,w} \geq l_{v,w}$ along with the standard constraints $x_{v,w} \leq u_{v,w}$. When can we claim that all basic feasible solutions are integral?*

Example 7.9 *Consider the problem where we are given a graph $G = (V, E)$, success probabilities $p_{v,w}$ on each edge (v, w), a source vertex s and a destination vertex t. Assume edge successes are independent. A path is successful if all the edges on the path are successful. Accordingly, the success probability, or the reliability, of a path (or a route) R is*

$$\prod_{(v,w)\in R} p_{v,w}.$$

The goal is to find the most reliable route from s to t. Assume all probabilities are between 0 and 1.

Since ln is an increasing function[7], maximizing the objective

$$\prod_{(v,w)\in R} p_{v,w}$$

gives the same route R as maximizing the objective

$$\ln\left(\prod_{(v,w)\in R} p_{v,w}\right).$$

By using the property that $\ln xy = \ln x + \ln y$, it is equivalent to maximize the objective

$$\sum_{(v,w)\in R} \left(\ln p_{v,w}\right),$$

or, to minimize the objective

$$\sum_{(v,w)\in R} \left(-\ln p_{v,w}\right).$$

This last objective function is just the shortest path objective with cost $c_{v,w} = -\ln p_{v,w}$. Thus, we can solve the most reliable path problem by

[7]The function $\ln(a)$, or the natural logarithm of a, is the number b such that $a = e^b$. We have $\ln 1 = 0; \ln 0 = -\infty; \ln xy = \ln x + \ln y$.

solving a shortest path problem using $-\ln p_{v,w}$ as the cost of edge (v, w). Since $p_{v,w} \leq 1$, we have $\ln p_{v,w} \leq 0$, or, $-\ln p_{v,w} \geq 0$. Hence the edge costs in the new shortest path problem are non-negative, and we will not get any -ve cycles. Thus, as long there is a path from s to t, solving the resulting shortest path problem as an LP (with $u =$ either ∞ or 1) will give a simple path as an optimum solution (assuming we use a typical commercial solver).

7.4 Max-flow

Consider a graph $G = (V, E)$ which has two special vertices s and t. Every edge has a capacity (but no cost). Instead of having demands on nodes, the goal is to send as much flow from s to t as possible, while respecting the capacity constraints. This is known as the max-flow problem, and has many applications. In order to model this as an LP, we use the same decision variables as for the min-cost-flow problem. The capacity constraints are the same as before:

$$\forall (v, w) \in E : x_{v,w} \leq u_{v,w},$$

and all variables are non-zero:

$$x \geq 0.$$

No node other than s or t can either produce or consume flow, so we have the following flow conservation constraints for all nodes v other than s and t:

$$\forall v \in V - \{s, t\} : \mathrm{OUT}_x(v) - \mathrm{IN}_x(v) = 0.$$

The goal is to maximize the net flow out of s, which must be the same as the net flow into t since all other nodes can not supply or consume flow. Hence, the objective function is:

$$\text{maximize } IN_x(t) - OUT_x(t),$$

which completes the LP formulation.

There is an interesting way to reduce max-flow to min-cost-flow. Keep the demands of all nodes as 0. Add a new edge from t to s and set its capacity to be infinite, and cost -1. For all the original edges in the graph G, keep the original capacity, and set the cost to 0. Since the new edge (t, s) has a cost of -1, and all other edges have a cost of 0, the objective function of the min-cost-flow problem is equivalent to maximizing the flow on edge (t, s). Since the demands are 0, any flow on the edge (t, s) must then be shipped from s to t through the original network to satisfy the conservation constraints; this is the original max-flow objective.

This allows us to claim that *if all the edge capacities are integral, then all basic feasible solutions to a max-flow-problem are integral.*

Exercise 7.10 *Suppose there are k special nodes $s_1, s_2, \ldots, s_k$ in a graph and j other special nodes $t_1, t_2, \ldots, t_j$. The goal is to maximize the total flow going from the nodes $s_1, \ldots, s_k$ to $t_1, \ldots, t_j$. How will you model this as a standard max-flow problem with only one source node and only one terminal node?*

8 Duality

Duality is a central concept in the theory of optimization. Duals of linear programs play an important role in understanding, using, and solving linear programs. Details are in VRM4; in this section, we will merely provide additional illustrative examples.

8.1 The knapsack problem revisited: duality

Suppose a crime syndicate wants to buy out the thief. They offer to pay the thief a price y_1 for the gold, a price y_2 for the diamonds, a price y_3 for the silver, and a price y_4 per lb for the knapsack[8]. But the thief can use 2 lbs of knapsack capacity and all her gold to generate a profit of 5 units, so $2y_4 + y_1$ should be at least 5. Similarly, $3y_4 + y_2 \geq 20$ and $4y_4 + y_3 \geq 3$. The syndicate would like to minimize the total price it pays, i.e. minimize $y_1 + y_2 + y_3 + 4y_4$. Also, the prices should be non-negative, otherwise the thief will hold on to that resource (gold, diamond, silver, or knapsack capacity). This gives us the following LP which the syndicate can use to decide the minimum price:

$$\begin{array}{rl} \text{minimize} & y_1 + y_2 + y_3 + 4y_4 \\ \text{subject to:} & \\ & \begin{array}{cccrcl} y_1 + & & & 2y_4 & \geq & 5 \\ & y_2 + & & 3y_4 & \geq & 20 \\ & & y_3 + & 4y_4 & \geq & 3 \\ & & & y & \geq & 0. \end{array} \end{array}$$

[8]The Excel spreadsheet names the variables a little differently, with y_1 being the price for knapsack capacity.

Try solving this linear program in Excel. It turns out that $y = \langle 0, 12.5, 0, 2.5\rangle$ is an optimum solution, giving an optimum value of 22.5, or, the same as the optimum value of the previous LP.

In retrospect, the connection is not really surprising. What is the minimum total price the thief would settle for? Clearly 22.5, since this is how much the thief can make by filling her knapsack and walking away from any deal being offered by the syndicate. This is an example of duality. The variables y_1, y_2, y_3, y_4 are called dual variables or dual prices. Duality is a very important concept in economics; in fact prices in the real world can often best be understood as dual variables.

Recall that N is the number of goods, which gave N variables and $N+1$ constraints in the original LP formulation of the general knapsack problem. Hence, the dual of the knapsack problem has $N+1$ variables $y_1, y_2, \dots y_{N+1}$, and N constraints labeled $x_1, x_2, \dots, x_N$. The dual is:

$$\begin{array}{ll} \text{minimize} & \left(\sum_{i=1}^{N} y_i\right) + W y_{N+1} \\ \text{subject to:} & \\ & \begin{array}{rrcl} \forall i \in \{1,\dots,N\}: & y_i + w_i y_{N+1} & \geq & v_i \\ & y & \geq & 0. \end{array} \end{array}$$

8.2 Dual prices as sensitivities

Try increasing the knapsack capacity by 0.1 in the above knapsack example. The optimum solution changes from 22.5 to 22.75. The ratio of change in the optimum solution to a very small (i.e., infinitesimally small) change in the right hand side of a constraint (assuming all the variables are to the left and the constant term is on the right) is known as the *sensitivity*. In this case, the sensitivity to knapsack capacity is $(22.75 - 22.5)/0.1 = 2.5$. This is not surprising, since any extra knapsack capacity (assuming the extra amount is small) is used for additional gold. Since gold weighs 2lbs, additional capacity used for gold gives us an additional profit of \$2.5 per unit of knapsack capacity.

What is surprising is that the dual variable y_4 corresponding to the knapsack capacity constraint has value 2.5. In general, a dual variable's value in an optimum dual solution is equal to the sensitivity of the primal to the corresponding constraint in the primal. In order to understand this, consider a primal linear program in standard form:

$$\text{maximize} \qquad c^T \cdot x$$

subject to:

$$\begin{array}{rcl} Ax & \leq & b \\ x & \geq & 0 \end{array}$$

and its dual:

$$\begin{array}{rl} \text{minimize} & b^T \cdot y \\ \text{subject to:} & \end{array}$$

$$\begin{array}{rcl} A^T y & \geq & c \\ y & \geq & 0. \end{array}$$

Assume for simplicity that the primal and the dual both have unique optimum solutions. Focus on the optimum solution (say y^*) for the dual. Since the optimum solution is unique, it must be a basic feasible solution. Imagine changing the right hand side of the i-th constraint in the primal slightly: i.e. suppose c_i is changed to $c_i + \delta_i$ where δ_i is infinitesimally small. For the dual, this only changes the objective function (since the c_i's only occur in the objective function in the dual) and hence does not impact the set of basic feasible solutions. For an infinitesimally small change in the objective, the same basic feasible solution will still remain optimum[9]. Hence, the change in the optimum objective value for the dual will be $\delta_i y_i^*$. By strong duality, this will also be the change in the primal. Hence, the sensitivity of the primal to the i-th constraint is $\delta_i y_i^* / \delta_i = y_i^*$.

If you solve a linear program using Excel, you get a sheet called "sensitivity": these are essentially the dual variables. Thus, we now have another concrete connection between the primal and the dual. This Excel sheet also has a "range". As we change the right hand side of a constraint, we are also changing the dual objective function, and when this change is large enough, some other basic feasible solution could become optimum. At this point, the sensitivities of the primal will change; the "range" gives the maximum amount by which we can change the right hand side of a constraint without changing the corresponding dual solution and hence the sensitivities.

If the dual has multiple *optimum* basic feasible solutions, at least one of the ranges will be 0, and you should interpret the sensitivities (or *shadow prices* as they are sometimes called) with caution. For an illustrative example, see the sensitivity report you obtain when you set the knapsack capacity to 5 in the above problem and then solve the primal.

[9]Since y^* is the *unique* optimum solution, it must be better than all other basic feasible solutions by at least some small positive amount, and an infinitesimally small change in the objective can not make another basic feasible solution optimum.

8.3 The dual of the shortest path problem

Consider a linear program of the form:

$$\begin{array}{rrcl} \text{minimize} & c^T \cdot x & & \\ \text{subject to:} & & & \\ & Ax & = & b \\ & x & \geq & 0. \end{array}$$

Suppose the LP has K constraints and J variables. We will reduce this to the standard primal form, take the dual, and simplify. We have had some practice reducing LPs to standard form, so we will go this through quickly. Define

$$E = \begin{pmatrix} A \\ -A \end{pmatrix}, f = \begin{pmatrix} b \\ -b \end{pmatrix}, \text{ and } g = -c.$$

Now the LP becomes

$$\begin{array}{rrcl} -\text{maximize} & g^T \cdot x & & \\ \text{subject to:} & & & \\ & Ex & \leq & f \\ & x & \geq & 0. \end{array}$$

We now have $2K$ constraints and J variables, and the dual is

$$\begin{array}{rrcl} -\text{minimize} & f^T \cdot z & & \\ \text{subject to:} & & & \\ & E^T z & \geq & g \\ & z & \geq & 0. \end{array}$$

The dual has $2K$ variables, $z_1, z \dots z_{2K}$. We will refer to the first K as $p_1 \dots p_K$ and the next K as $q_1 \dots q_K$. The constraint $E^T z \geq g$ is now equivalent to $A^T(p-q) \geq g$ and the objective function $f^T z$ is now equivalent to $b^T(p-q)$. Since p, q always occur together as $p-q$, we might as well define a new set of variables $y_1, \dots, y_K$ such that $y_i = q_i - p_i$. While the variables z, and hence the variables p, q, were constrained to be non-negative, the variables y are unconstrained. Replacing $f^T z$ by $b^T(-y)$, $E^T z$ by $A^T \cdot (-y)$, g by $-c$ and removing the non-negativity constraints, we have the following equivalent dual LP:

$$\begin{array}{rrcl} -\text{minimize} & c^T \cdot (-y) & & \\ \text{subject to:} & & & \\ & A^T(-y) & \geq & -b, \end{array}$$

which simplifies to:

$$\begin{array}{lrcl} \text{maximize} & c^T y & & \\ \text{subject to:} & & & \\ & A^T y & \leq & b. \end{array}$$

What does all this have to do with the shortest path problem? Write the shortest path problem as:

$$\begin{array}{llrcl} \text{minimize} & \sum_{(v,w)\in E} c_{v,w} x_{v,w} & & & \\ \text{subject to:} & & & & \\ & \forall v \in V - \{s,t\}: & \text{IN}_x(v) - \text{OUT}_x(v) & = & 0 \\ & & \text{IN}_x(t) - \text{OUT}_x(t) & = & 1 \\ & & \text{IN}_x(s) - \text{OUT}_x(s) & = & -1 \\ & & x & \geq & 0. \end{array}$$

This linear program is now in the form with which we started this subsection, and hence we can read off its dual using the form that we derived. The primal linear program has M variables, one for each edge (M is the number of edges) and N constraints, one for each node. Its dual will have M constraints (one for each edge) and N variables (one for each node). We will label the variables in the dual as y_v, where $v \in V$. The dual constraint corresponding to edge (v,w) is simple: $y_w - y_v \leq c_{v,w}$. The objective function is $y_t - y_s$. The dual is:

$$\begin{array}{llrcl} \text{maximize} & y_t - y_s & & & \\ \text{subject to:} & & & & \\ & \forall (v,w) \in E: & y_w - y_v & \leq & c_{v,w}. \end{array}$$

This is a starkly simple form! Let us interpret this *informally.* First, observe that in the primal, the constraint $IN(s) - OUT(s) = 0$ is redundant, since it can be obtained by adding all the other conservation constraints. Removing this constraint from the primal is equivalent to setting $y_s = 0$ in the dual, which is what we will implicitly assume from now on. Now, suppose we change the right hand side of the conservation constraint corresponding to node v by an infinitesimally small amount δ_v. This is equivalent to saying that the demand of node v has been increased by δ_v. How will this demand get satisfied? Since there are no capacities in the primal, this demand will get satisfied by *sending flow from s to v along a shortest path from s to v*; this will incur a cost of δ_v times the cost of the shortest path from s to v. Hence we expect the dual variables y_v to represent the shortest path distance between s and v.

What about the constraints $\forall (v,w) \in E : y_w - y_v \leq c_{v,w}$? This corresponds to enforcing triangle inequality. If the shortest path length from s to v is y_v, then the shortest path length y_w from s to w can be at most $y_v + c_{v,w}$: having arrived at v, we need to pay at most $c_{v,w}$ extra to get to w.

This dual linear program gives rise to the following simple algorithm for finding shortest paths:

Ford's algorithm

1. Set $y_s = 0$ and all the other y_v's to ∞ (i.e. some very large number, for example $1 + \sum_{(v,w)\in E)} |c_{v,w}|$)
2. while there exists an edge (v,w) such that $y_w > y_v + c_{v,w}$ set $y_w = y_v + c_{v,w}$.

The algorithm is guaranteed to terminate if there are no negative cost cycles. An edge (v,w) such that $y_w > y_v + c_{v,w}$ is called a *violated edge*; fixing the edge by setting $y_w = y_v + c_{v,w}$ is called *relaxation.* This is the canonical algorithm used for most shortest path applications with the main variation being in the method to decide which edge is relaxed when there are multiple violated edges.

Exercise 8.1 *Manually simulate Ford's algorithm for the shortest path example in figure 1.*

Thus, we see how linear program duality can have a great impact in obtaining efficient algorithms for important problems.

9 Dynamic Programming – a (very) brief introduction

The term dynamic programming is used in multiple ways in optimization. In this class, we will focus on a narrow chunk of dynamic programming, where (a) The overall problem can be decomposed into a tractable number of subproblems, and (b) Each subproblem can be solved efficiently if all subproblems of smaller "size" have been solved. Part (a), which involves only writing down notation and definitions, is by far the hardest part. Part (b) usually follows, and once we have both parts (a) and (b), solving using Excel or a programming language is generally trivial. We will illustrate this technique for problems which can be solved using Excel (i.e. where the subproblems are at most 2-dimensional).

9.1 Longest Common Subsequence, aka, Are you a man or a mouse?

The field of computational genomics involves looking at the human (or another species') genome as a sequence of A, C, T, G (the four DNA bases) and using computation on these sequences to determine genetic functionality, similarity between species, similarity between individuals, disease markers, etc. It is a field which is revolutionizing biology, and dynamic programming is a key technique.

Rather than survey the entire field, we will discuss a representative problem called "Longest Common Subsequence". Here we are given two sequences of bases $P = \langle p_1, p_2, \ldots, p_M \rangle$ and $Q = \langle q_1, q_2, \ldots, q_N \rangle$, of lengths M and N respectively. Remember that a sequence is different from a set in that the bases are ordered, i.e. they occur in sequence.

A subsequence of a sequence is obtained by deleting an arbitrary number of elements from the sequence; the remaining elements form the subsequence. The elements of a subsequence must be in the same order as the original sequence, but need not be contiguous in the original sequence. Also, the elements in a sequence need not be unique. Given the sequence $\langle AAACCTTTAAGGGA \rangle$ the following are all subsequences: $\langle ACTG \rangle$, $\langle AATGGA \rangle$, $\langle TTT \rangle$, $\langle \rangle$, the last of these being the special sequence called the empty sequence, of length 0. But $\langle TC \rangle$ is not a subsequence. Our goal is to find the length of the longest common subsequence of P and Q. This is clearly an optimization problem: notice the term "longest".

This problem is of key important to phylogeny, the branch of genomics/evolution that attempts to make an evolution graph showing how species could have evolved. Scientists use the longest common subsequence problem (and its variants) to determine how similar current or archaeologically obtained genomes of various species are, which allows them to deduce evolutionary distances between species. Variants of this problem are also of great importance in discovering genetic markers for disease, physical traits, and behavior. There are several large supercomputer clusters whose main purpose is to run a tool called BLAST which uses variants of the lcs problem to discover similarity between genetic material.

Without further ado, we will proceed to step (a). We define $L(i, j)$ to be the length of the longest common subsequence between the first i elements of P and the first j elements of Q. The first i elements of P will be denoted $P[1 \ldots i]$ the first j elements of Q will be denoted $Q[1 \ldots j]$.

Notice that part (a) is very simple to write down, but very perplexing to discover the first time you see a problem. At the same time, it can be very exhilarating to encounter a complex problem, just define the right subproblems, and watch the problem solve itself, as this one will. We will define $L(0, j) = L(i, 0) = 0$ for all i, j. The number we are interested in is $L(M, N)$.

Suppose $P(i) = Q(j)$. If the last bases are the same, there is no advantage to not putting them both in the common subsequence, and we have $L(i, j) = 1 + L(i-1, j-1)$. If $P(i) \neq Q(j)$ then every common subsequence of $P[1 \dots i]$ and $Q[1 \dots j]$ must involve deleting either the i-th element of P or the j-th element of Q (or both). Since we do not know in advance which of the two will be deleted, we can simply set $L(i, j) = \max\{L(i-1, j), L(i, j-1)\}$, and we are done. Notice that to compute $L(i, j)$ we only need values of L with a smaller i or a smaller j so the "size" requirement of part (b) is satisfied. To summarize:

1. Define $L(i, 0) = L(0, j) = 0$. This correspond to making an $(M+1) \times (N+1)$ table in Excel and making the first row and the first column 0.

2. Fill the rest of the table using the formula

$$\begin{aligned} L(i,j) &= 1 + L(i-1, j-1) && \text{if } P(i) = Q(j) \\ &= \max\{L(i-1, j), L(i, j-1)\} && \text{other wise.} \end{aligned}$$

   It is important to note that the above formula will only be used when $i \geq 1$ and $j \geq 1$ and hence all the L values used have been previously defined.

If your Excel tables are properly set up, the values $P(i), Q(j), L(i-1, j), L(i, j-1), L(i, j)$ are all easy to locate using Excel style addressing (using \$'s to fix the rows/columns where you look up P, Q). Just type in the above formula in the cell corresponding to $L(1, 1)$ and copy/paste the formula into the rest of the table. Since the formula only refers to cells of a smaller "size" i.e. smaller i or j, there will be no cyclic definitions and Excel will fill the whole table. You can then read off the value of $L(M, N)$. The Excel file lcs.xls provides an example. It is clear that you can solve this problem easily in any high level programming language. We will now see another interesting application of lcs:

Example 9.1 *You are given a sequence P of numbers. Find the largest non-decreasing subsequence of P.*

The above problem can be solved by sorting P; let Q be the sorted sequence. The lcs of P and Q solves the problem.