Presentation for use with the textbook, Algorithm Design and Applications, by M. T. Goodrich and R. Tamassia, Wiley, 2015

Linear Programming

Brooklyn Bridge showing painters on suspenders. By Eugene de Salgnac, October 7, 1914. From NYC Municipal Archives. Public domain image.

Optimization Problems

- Optimization problems are common in the real world
  - Ex. Maximize profit
  - Ex. Minimize error, minimize cost
- In general, such problems have two components:
  - A set of contraints that must be satisfied
  - An objective function to maximize or minimize, subject to the contraints

Example

- Web company want to buy new servers to replace outdated ones, and has two choices:
- Standard model
  - Costs $400
  - Uses 300W of power
  - Takes two shelves of server rack
  - Can handle 1000 hits/min
- Cutting-edge model
  - Costs $1600
  - Uses 500W of power
  - Takes one shelf of server rack
  - Can handle 2000 hits/min

Example (cont.)

- Budget is $36,800
- Company has 44 shelves of server space
- Company has 12,200 Watts of power
- Question: How many units of each server should company purchase in order to maximize the number of hits it can serve every minute?
- Let x_1 = number of standard server
- Let x_2 = number of cutting-edge server
- Goal: maximize $1000x_1 + 2000x_2$ subject to contraints

Constraints:

- Cost: $400x_1 + 1600x_2 \leq 36{,}800$
- Shelves: $2x_1 + x_2 \leq 44$
- Power: $300x_1 + 500x_2 \leq 12200$
- $x_1 \geq 0, \quad x_2 \geq 0$

Problem Summary

Therefore, this optimization problem can be summarized as follows:

$$\begin{aligned} \text{maximize:} \quad & z = 1000x_1 + 2000x_2 \\ \text{subject to:} \quad & 400x_1 + 1600x_2 \leq 36800 \\ & 2x_1 + x_2 \leq 44 \\ & 300x_1 + 500x_2 \leq 12200 \\ & x_1, x_2 \geq 0, \end{aligned}$$

Solving optimization problems that have the above general form is known as *linear programming*.

Translating Problems into Linear Programs

- Determine the variables of the problem
- Find the quantity to optimize, and write it in terms of the variables
- Find all the constraints on the variables and write equations or inequalities to express these constraints
  - Be sure to include any implicit constraints on the range of values of variables
  - Make sure all equations are *linear*
- We will study the *simplex method*

Formulating the Problem

Standard Form

Recall that a function, f, is a ***linear function*** in the variables, $x_1, x_2, \ldots, x_n$, if it has the following form:

$$f(x_1, x_2, \ldots, x_n) = a_1x_1 + a_2x_2 + \cdots + a_nx_n = \sum_{i=1}^{n} a_ix_i,$$

for some real numbers, $a_1, a_2, \ldots, a_n$, which are called ***coefficients*** or ***weights***.

A ***linear program*** in ***standard form*** is an optimization problem with the following form:

$$\begin{aligned} \text{maximize:} \quad & z = \sum_{i \in V} c_i x_i \\ \text{subject to:} \quad & \sum_{j \in V} a_{ij} x_j \leq b_i \text{ for } i \in C \\ & x_i \geq 0 \text{ for } i \in V \end{aligned}$$

where V indexes over the set of variables and C indexes over the set of constraints.

- The function to maximize is called the **objective function** and the inequalities are called **constraints**.

Matrix Notation

A linear function can be expressed as a dot product:

$$\sum_{i=1}^{n} a_i x_i = \vec{a} \cdot \vec{x}, \qquad \begin{aligned} \vec{a} &= (a_1, \ldots, a_n), \\ \vec{x} &= (x_1, \ldots, x_n). \end{aligned}$$

Notice that $\vec{a}$ is a vector of numbers while $\vec{x}$ is a vector of variables. Using dot products, we can express the standard form more compactly:

$$\begin{aligned} &\text{maximize:} && \vec{c} \cdot \vec{x} \\ &\text{subject to:} && \vec{a}_1 \cdot \vec{x} \leq b_1 \\ & && \vec{a}_2 \cdot \vec{x} \leq b_2 \\ & && \vdots \\ & && \vec{a}_m \cdot \vec{x} \leq b_m. \end{aligned}$$

Compact Matrix Notation

In fact, we can express it even more compactly, by letting A be the matrix where the ith row is vector $\vec{a}_i$. In other words, A is the $m \times n$ matrix whose ijth entry is a_{ij}. Also let $\vec{b}$ be the vector with entries b_i. If we extend the meaning of the symbol $\leq$ to row-wise inequality, we can have a more succinct description of standard form as follows:

maximize:	$\vec{c} \cdot \vec{x}$
subject to:	$A\vec{x} \leq \vec{b}.$

For example, the inequalities

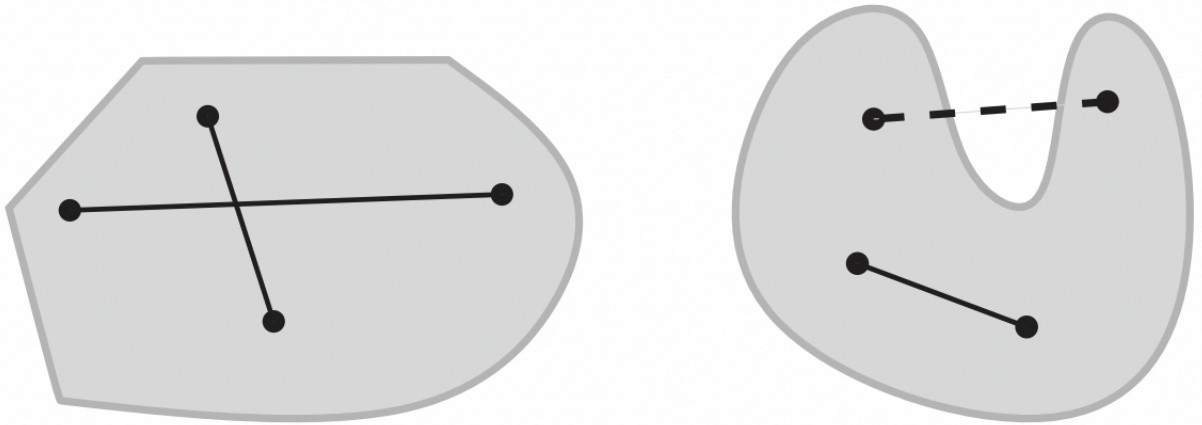
$$\begin{aligned} 2x + z &\leq 5 \\ x - 4y - 3z &\leq 1 \end{aligned}$$

can be written as

$$\begin{aligned} (2,0,1) \cdot (x,y,z) &\leq 5 \\ (1,-4,-3) \cdot (x,y,z) &\leq 1 \end{aligned} \quad \text{or} \quad \begin{pmatrix} 2 & 0 & 1 \\ 1 & -4 & -3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \leq \begin{pmatrix} 5 \\ 1 \end{pmatrix}.$$

Recall: Convex Region

A region is ***convex*** if any two points in the region can be joined by a line segment entirely in the region (see Figure 26.2). Intuitively, if a two-dimensional shape is convex, then a person walking on its boundary, assuming it has one, would always be making left turns, or always right turns. (See, also, Section 22.2.)

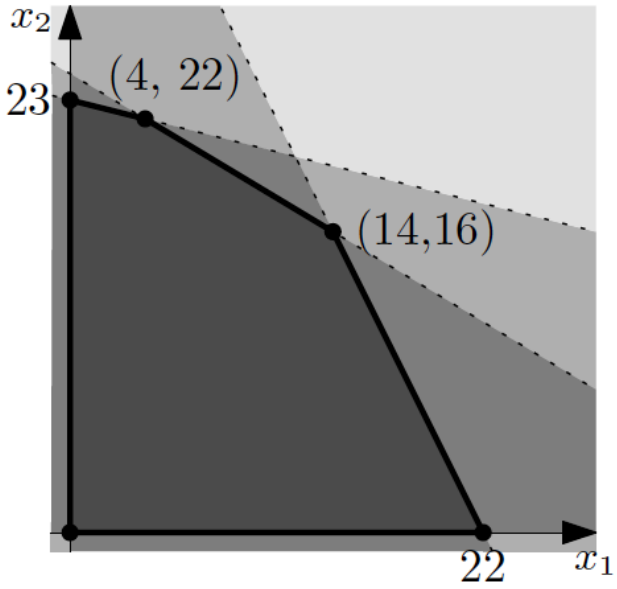


(a) (b)

Figure 26.2: (a) In a convex set, any segment line joining two points inside the set is also inside the set. (b) In a non-convex set, there are segments joining two points in the set that are not entirely in the set.

Geometry of Linear Programs

- Let us restrict ourselves to the two-dimensional case:
  - Each inequality constraint describes a half-plane, expressed as an infinite region to one side of a line.
  - So a point that is inside all of these half-planes is a point that satisfies all the constraints.
  - We call such a point a **feasible solution**.
  - Intersections of half-planes have the shape of a convex polygon (Section 22.2).
  - We call this set the **feasible region**.

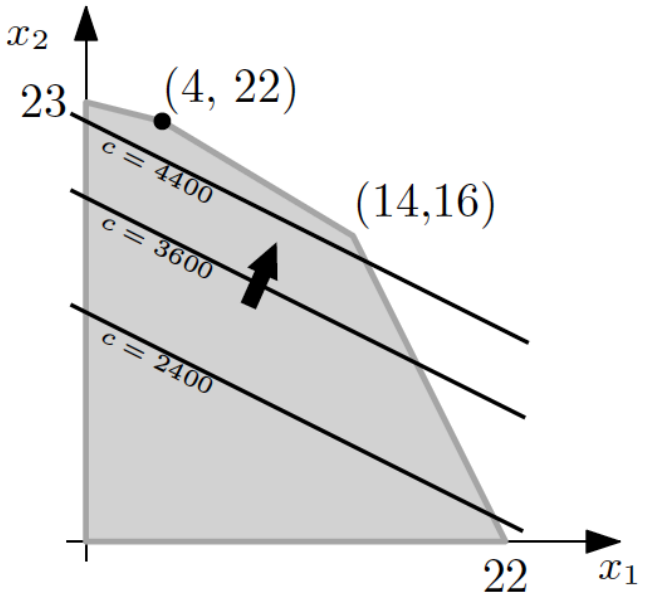


Optimization

- We are interested in a feasible solution that optimizes the objective function. We refer to this feasible solution an **optimal solution**.
- For example, consider the objective function

$$c = 1000x_1 + 2000x_2.$$

- Applying this to previous 2-dimensional constraints, optimization looks like a moving line that intersects the feasible region.



The Simplex Method

- To solve a linear program using the **simplex method**, we must first rewrite the problem in a format known as **slack form**.
- To convert a linear program from standard form into slack form, we rewrite each of the inequality constraints as an equivalent equality constraint.
- This is done with the introduction of new variables, called **slack variables**, which are nonnegative and measure the difference in the original inequality.

Slack Variables

- These are nonnegative variables that measure the difference between a quantity and its constraint.
  - This allows us to turn inequalities into equalities
- Example: Suppose we have the constraint $2x - 5y \leq 28$
- In slack form we can write this as $s = 28 - (2x - 5y)$
  - s is the slack variable here

Slack Variables

- NOTE: It would be nice to denote our slack variables as "s_i"
- But in the simplex method (to come) we often denote them as additional "x_i" variables.
- Ex. Our problem might have variables x_1 and x_2. If we need three slack variables to turn constraints into inequalities, we might add slack variables x_3, x_4, and x_5.
  - This can be confusing, so be aware of it!
  - Bottom line: some of the x_i are "original" to the problem and some are slack variables that have been added!

Slack Variables

- Bottom line: some of the x_i are ”original” to the problem and some are slack variables that have been added!
- In effect, the x_i are partitioned into two sets:
  - B: the “basic” (slack) variables, and
  - F: the “free” (original) variables

Slack Form

Formally, we say that linear program is in ***slack form*** if we seek to maximize a linear objective, z, subject to constraints that are either equality constraints involving a slack variable or are nonnegativity constraints, as follows:

$$\begin{aligned}
&\text{maximize:} && z = c_* + \sum_{j \in F} c_j x_j \\
&\text{subject to:} && x_i = b_i - \sum_{j \in F} a_{ij} x_j, \quad \text{for } i \in B \\
& && x_i \geq 0 \text{ for } 1 \leq i \leq m + n.
\end{aligned}$$

The sets B and F partition the x_i variables into ***basic variables*** and ***free variables***, respectively. That is, each equality constraint has a basic (slack) variable on the left-hand side and only free variables on the right-hand side. Thus, free variables only appear on the left-hand side of nonnegativity constraints. Taken together, the a_{ij} coefficients form a $m \times n$ matrix, A, where $n = |F|$ is the number of variables in the standard form, and $m = |B|$ is the number of constraints in the standard form. Incidentally, the minus sign in the equality constraints is needed so that the matrix A is the same in the slack form and standard form.

Example

Example 26.1: *Below we convert the linear program in standard form (left) into a slack form (right). In this particular slack form, the basic variables are slack variables, but this is not always the case.*

$$\begin{aligned} &\textit{maximize:} && z = x_1 + 2x_2 \\ &\textit{subject to:} && -3x_1 + 2x_2 \leq 3 \\ & && x_1 + x_2 \leq 2 \\ & && x_1 - x_2 \leq 1 \\ & && x_1, x_2 \geq 0 \end{aligned} \qquad \begin{aligned} &\textit{maximize:} && z = x_1 + 2x_2 \\ &\textit{subject to:} && x_3 = 3 + 3x_1 - 2x_2 \\ & && x_4 = 2 - x_1 - x_2 \\ & && x_5 = 1 - x_1 + x_2 \\ & && x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

In this slack form, the free variables have indices $F = \{1, 2\}$ *and the basic variables have indices* $B = \{3, 4, 5\}$.

Basic Solutions and Pivots

- We are interested in the basic solution of the slack form, which means we set all the free variables to zero and let the equality constraints determine the values of the basic variables.
- The simplex method works by rewriting the slack form until a basic solution becomes an optimal solution.
- The operation we use to rewrite the slack form is called a **pivot**, which takes a free variable and a basic variable and interchanges their roles by rewriting the equality constraints and objective function.

Note this is a correction. Text is wrong.

Example Pivot

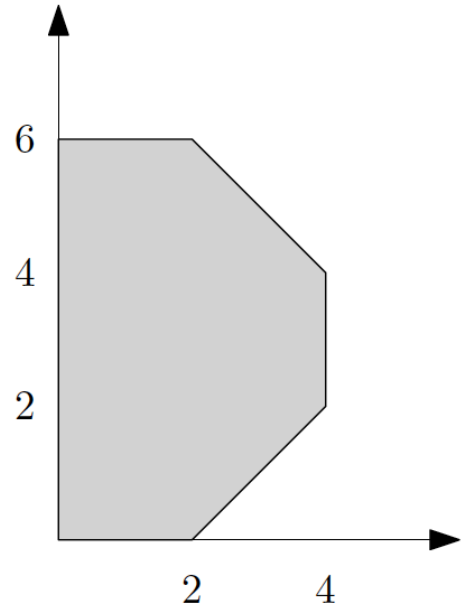
Example 26.2: *In this example, we perform a pivot where the free variable x_1 becomes basic and the basic variable x_5 becomes free. We perform this pivot by rewriting the equality constraint with x_5 on the left-hand side so that it has x_1 on the left-hand side. Then, we substitute this new equality constraint for x_1 in the old objective function and equality constraints to obtain a new objective function and new equality constraints involving only the free variables, x_2 and x_5, on the right-hand sides.*

maximize:	$z = x_1 + 2x_2$		maximize:	z	$=$	$1 + 3x_2 - x_5$
subject to:	$x_3 = 3 + 3x_1 - 2x_2$		subject to:	x_3	$=$	$6 + x_2 - 3x_5$
	$x_4 = 2 - x_1 - x_2$			x_4	$=$	$1 - 2x_2 + x_5$
	$x_5 = 1 - x_1 + x_2$			x_1	$=$	$1 + x_2 - x_5$
	$x_1, x_2, x_3, x_4, x_5 \geq 0$			$x_1, x_2, x_3, x_4, x_5 \geq 0.$		

The basic solution of the new slack form is $(1, 0, 6, 2, 0)$ and the objective value is 1. So, as a result of the pivot, we have increased the objective value from 0 to 1. Geometrically, we have started from the vertex $(0, 0)$, where the inequalities $x_1, x_2 \geq 0$ are tight (that is, they are satisfied by equality with the right-hand side), and we have moved to the vertex $(1, 0)$ where the inequalities $x_2, x_5 \geq 0$ are tight.

An Extended Example

$$
\begin{aligned}
\text{maximize:} \quad & z = 4x_1 + x_2 \\
\text{subject to:} \quad & x_2 \leq 6 \\
& x_1 + x_2 \leq 8 \\
& x_1 \leq 4 \\
& x_1 - x_2 \leq 2 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



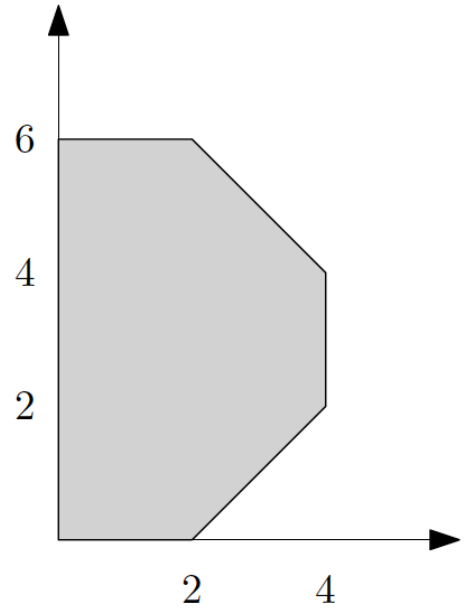
First, we rewrite the linear program into the initial slack form, introducing the slack variables x_3, x_4, x_5 and x_6.

$$
\begin{aligned}
\text{maximize:} \quad & z = 4x_1 + x_2 \\
\text{subject to:} \quad & x_3 = 6 - x_2 \\
& x_4 = 8 - x_1 - x_2 \\
& x_5 = 4 - x_1 \\
& x_6 = 2 - x_1 + x_2 \\
& x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
\end{aligned}
$$

Free Variables: $F = \{1, 2\}$
Basic Variables: $B = \{3, 4, 5, 6\}$
Basic Solution: $(0, 0, 6, 8, 4, 2)$
Objective Value: $c_* = 0$

An Extended Example

$$
\begin{aligned}
\text{maximize:} \quad & z = 4x_1 + x_2 \\
\text{subject to:} \quad & x_2 \leq 6 \\
& x_1 + x_2 \leq 8 \\
& x_1 \leq 4 \\
& x_1 - x_2 \leq 2 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



First, we rewrite the linear program into the initial slack form, introducing the slack variables x_3, x_4, x_5 and x_6.

$$
\begin{aligned}
\text{maximize:} \quad & z = 4x_1 + x_2 \\
\text{subject to:} \quad & x_3 = 6 - x_2 \\
& x_4 = 8 - x_1 - x_2 \\
& x_5 = 4 - x_1 \\
& x_6 = 2 - x_1 + x_2 \\
& x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
\end{aligned}
$$

Free Variables: $F = \{1, 2\}$
Basic Variables: $B = \{3, 4, 5, 6\}$
Basic Solution: $(0, 0, 6, 8, 4, 2)$
Objective Value: $c_* = 0$

$$x_1 = 2 + x_2 - x_6$$

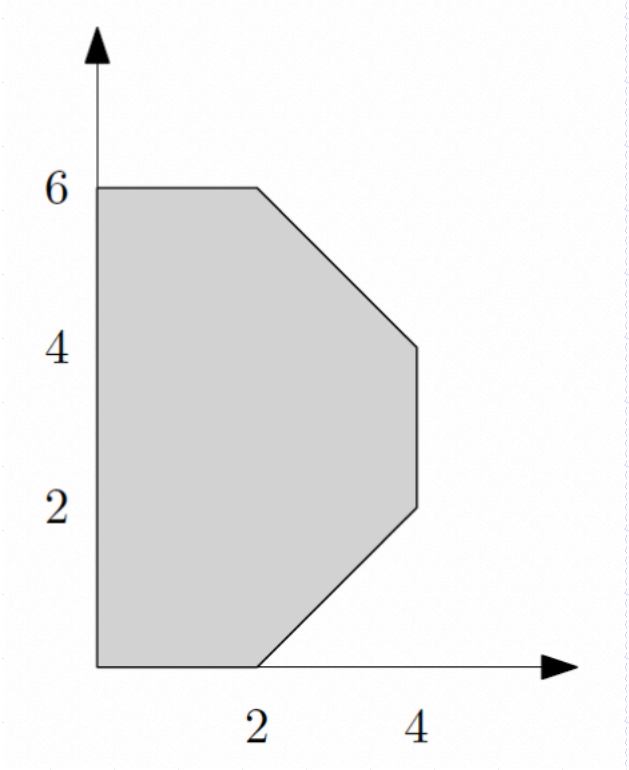
An Extended Example, step 2

Next, we look at the objective function and notice that increasing either x_1 or x_2 will increase the objective value. We choose to raise the value of x_1 as it has the largest coefficient. This greedy strategy does not necessarily improve runtime, by the way, but it is nevertheless often a useful choice in practice. While keeping x_2 fixed at zero, we increase x_1 as far as possible. The nonnegativity constraints of the basic variables impose the constraints $x_1 \leq \infty$ from x_3, $x_1 \leq 8$ from x_4, $x_1 \leq 4$ from x_5 and $x_1 \leq 2$ from x_6. The tightest of these constraints is $x_1 \leq 2$. So we pivot x_1 and x_6, setting $x_1 = 2$ and $x_6 = 0$ in the basic solution. This produces a new but equivalent slack form.

$$
\begin{aligned}
\text{maximize:} \quad & z = 8 + 5x_2 - 4x_6 \\
\text{subject to:} \quad & x_1 = 2 + x_2 - x_6 \\
& x_3 = 6 - x_2 \\
& x_4 = 6 - 2x_2 + x_6 \\
& x_5 = 2 - x_2 + x_6 \\
& x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
\end{aligned}
$$

Free Variables: $F = \{2, 6\}$
Basic Variables: $B = \{1, 3, 4, 5\}$
Basic Solution: $(2, 0, 6, 6, 2, 0)$
Objective Value: $c_* = 8$

An Extended Example, step 2



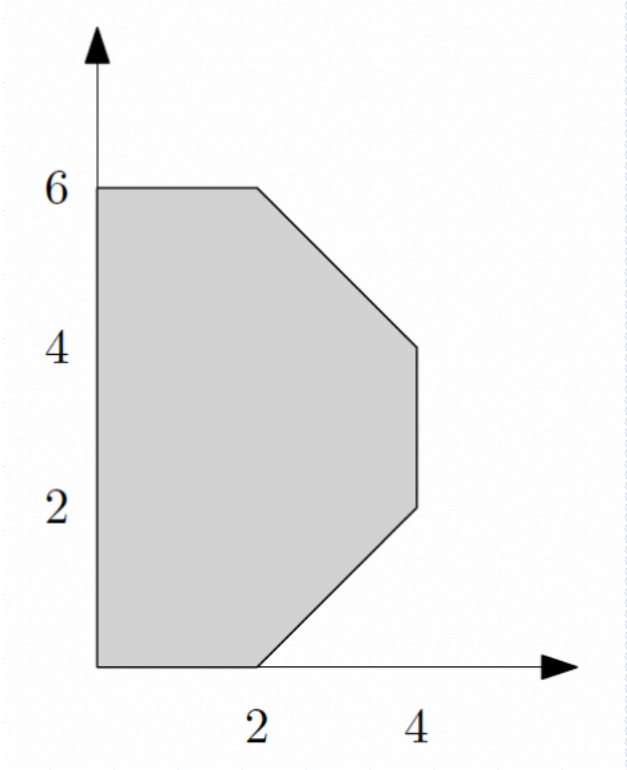
An Extended Example, step 3

Since the coefficient on x_6 is negative, raising it from zero will only decrease the objective function. For this reason, we keep x_6 fixed at zero and raise x_2 until we hit the first constraint. The constraints on x_2 are: $x_2 \leq \infty$, $x_2 \leq 6$, $x_2 \leq 3$, and $x_2 \leq 2$. The tightest of these constraints is $x_2 \leq 2$ coming from the basic variable x_5. So we set $x_2 = 2$ and $x_5 = 0$ by pivoting x_2 and x_5. This yields a new slack form.

$$
\begin{aligned}
\text{maximize:} \quad & z = 18 - 5x_5 + x_6 \\
\text{subject to:} \quad & x_1 = 4 - x_5 \\
& x_2 = 2 - x_5 + x_6 \\
& x_3 = 4 + +x_5 - x_6 \\
& x_4 = 2 + 2x_5 - x_6 \\
& x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
\end{aligned}
$$

Free Variables: $F = \{5, 6\}$
Basic Variables: $B = \{1, 2, 3, 4\}$
Basic Solution: $(4, 2, 4, 2, 0, 0)$
Objective Value: $c_* = 18$

An Extended Example, step 2



An Extended Example, step 4

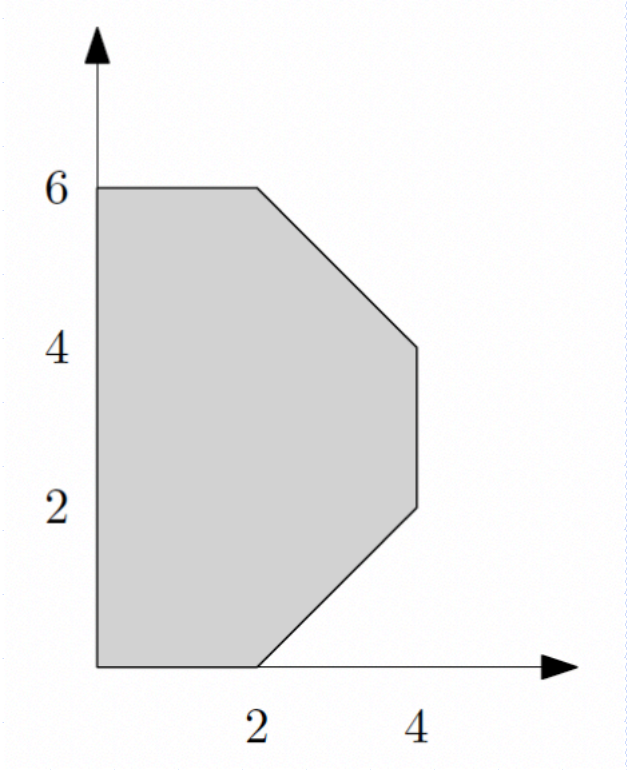
In this slack form, the coefficient of x_5 is negative. So we keep x_5 fixed at zero and increase x_6. The tightest constraint is $x_6 \leq 2$ coming from the basic variable x_4. So we pivot x_4 and x_6, yielding the next slack form.

$$
\begin{aligned}
\text{maximize:} \quad & z = 20 - 3x_5 - x_4 \\
\text{subject to:} \quad & x_1 = 4 - x_5 \\
& x_2 = 4 - x_4 + x_5 \\
& x_3 = 2 + x_4 - x_5 \\
& x_6 = 2 + 2x_5 - x_4 \\
& x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
\end{aligned}
$$

Free Variables: $F = \{4, 5\}$
Basic Variables: $B = \{1, 2, 3, 6\}$
Basic Solution: $(4, 4, 2, 0, 0, 2)$
Objective Value: $c_* = 20$

Now something interesting has happened! Both of the variables in the objective function have negative coefficients, so increasing either of them would decrease the objective function. So we conclude that we must be at the optimum and stop. From the basic solution we see that $x_1 = 4$, $x_2 = 4$ and $c_* = 20$, which is the solution to our original problem.

An Extended Example, step 2



The Simplex Algorithm

- We assume the input is given in slack form and that the basic solution is feasible.

SimplexMethod(A, b, c, c_*, F, B) :

while there exists $j \in F$ with $c_j > 0$ **do**
  $r \leftarrow \arg\max_{j \in F} c_j$
  for $i \in B$ **do**
    $k_i \leftarrow$ (**if** $a_{ir} \neq 0$ **then** $-b_i/a_{ir}$ **else** ∞)
  $s \leftarrow \arg\min_{i \in B} k_i$
  if $k_s = \infty$ **then**
    return unbounded exception
  else
    Pivot x_r and x_s.
return (A, b, c, c_*, F, B)

Another Simplex Example

Example 26.4: *Consider the following linear program, written in standard form:*

$$\begin{aligned} &\textit{maximize:} && z = x_1 + 2x_2 \\ &\textit{subject to:} && -x_1 + x_2 \leq 3 \\ & && x_1 + 3x_2 \leq 13 \\ & && x_1 - x_2 \leq 1 \end{aligned}$$

To solve this linear program using the simplex method, we first we rewrite the linear program in slack form, introducing the slack variables, x_3, x_4 *and* x_5.

$$\begin{aligned} &\textit{maximize:} && z = x_1 + 2x_2 \\ &\textit{subject to:} && x_3 = 3 + x_1 - x_2 \\ & && x_4 = 13 - x_1 - 3x_2 \\ & && x_5 = 1 - x_1 + x_2 \end{aligned}$$

We then choose to increase x_2, *as it has the largest coefficient in the objective function. The most restrictive constraint is given by* x_3. *So we pivot* x_2 *and* x_3, *yielding the following new slack form with objective value,* $c_* = 6$.

$$\begin{aligned} &\textit{maximize:} && z = 6 + 3x_1 - 2x_3 \\ &\textit{subject to:} && x_2 = 3 + x_1 - x_3 \\ & && x_4 = 4 - 4x_1 + 3x_3 \\ & && x_5 = 4 - x_3 \end{aligned}$$

Next, we increase x_1, *as it has the largest coefficient in the objective function. The most restrictive constraint is* x_4. *So we pivot* x_1 *and* x_4, *which yields the following slack form with objective value,* $c_* = 9$.

$$\begin{aligned} &\textit{maximize:} && z = 9 + 0.25x_3 - 0.25x_4 \\ &\textit{subject to:} && x_1 = 1 + 0.75x_3 - 0.25x_4 \\ & && x_2 = 4 - 0.25x_3 - 0.25x_4 \\ & && x_5 = 4 - x_3 \end{aligned}$$

This time, we increase x_3. *Its most restrictive constraint comes from* x_5. *So we pivot* x_3 *with* x_5. *We get the new slack form below, with objective value,* $c_* = 10$.

$$\begin{aligned} &\textit{maximize:} && z = 10 - 0.25x_4 - 0.25x_5 \\ &\textit{subject to:} && x_1 = 4 - 0.25x_4 - 0.75x_5 \\ & && x_2 = 3 - 0.25x_4 + 0.25x_5 \\ & && x_3 = 4 - x_5 \end{aligned}$$

Now that all the coefficients of the objective function are negative, we see that the optimal value for this linear program is 10, with $x_1 = 4$ *and* $x_2 = 3$.

Bellman-Ford Algorithm

- Works even with negative-weight edges
- Must assume directed edges (for otherwise we would have negative-weight cycles)
- Iteration i finds all shortest paths that use i edges.
- Running time: O(nm).
- Can be extended to detect a negative-weight cycle if it exists
  - We won't discuss that

Bellman-Ford Algorithm: Details

Algorithm BellmanFordShortestPaths($\vec{G}, v$):

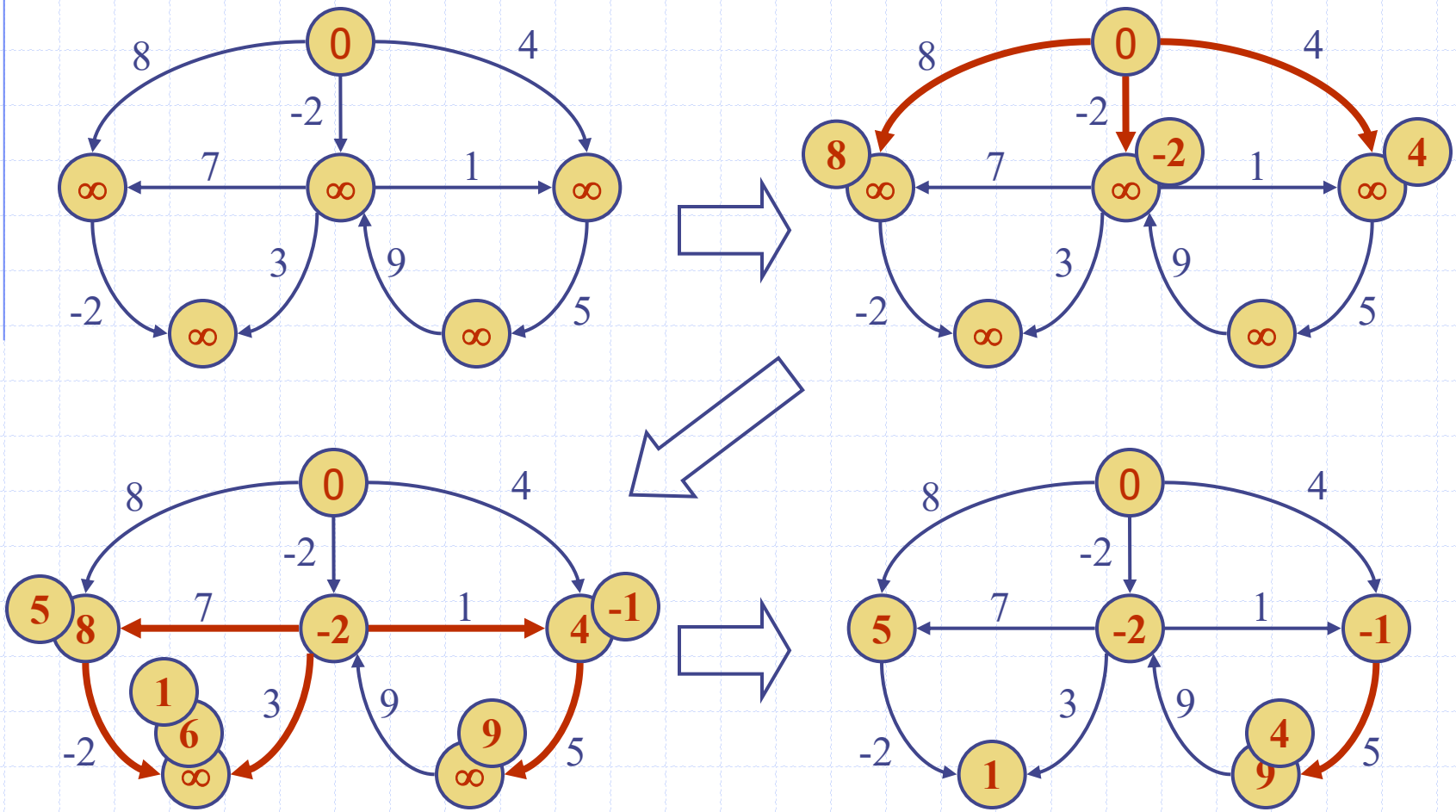
Input: A weighted directed graph $\vec{G}$ with n vertices, and a vertex v of $\vec{G}$

Output: A label $D[u]$, for each vertex u of $\vec{G}$, such that $D[u]$ is the distance from v to u in $\vec{G}$, or an indication that $\vec{G}$ has a negative-weight cycle

$D[v] \leftarrow 0$
for each vertex $u \neq v$ of $\vec{G}$ **do**
    $D[u] \leftarrow +\infty$
for $i \leftarrow 1$ to $n - 1$ **do**
    for each (directed) edge (u, z) outgoing from u **do**
        // Perform the ***relaxation*** operation on (u, z)
        if $D[u] + w((u, z)) < D[z]$ **then**
            $D[z] \leftarrow D[u] + w((u, z))$
if there are no edges left with potential relaxation operations **then**
    return the label $D[u]$ of each vertex u
else
    return "$\vec{G}$ contains a negative-weight cycle"

Bellman-Ford Example

Nodes are labeled with their D[v] values



Application of Linear Programming to Shortest Paths

- In the shortest-path problem, we have a source, s, and target, t, such that each edge, (u, v), has a weight, d(u, v), that represents the distance between two locations, u and v. The goal is to find the shortest path between the source, s, and target, t.
- We can formulate a linear program for this problem based on the setup and termination conditions for the Bellman-Ford algorithm.
- We define for every vertex, v, a variable, d_v, which should represent the shortest distance between s and v. The initialization, ds = 0, now becomes one of our constraints.
- To pinpoint the correct value for d_t, we use the termination condition of the Bellman-Ford algorithm—namely, that the triangle inequality holds for every edge. We want d_t to be the largest value that satisfies these conditions. Thus, the corresponding linear program is the following:

$$\begin{aligned} &\text{maximize:} && d_t \\ &\text{subject to:} && d_s = 0 \\ & && d_v \leq d_u + d(u,v), \quad \text{for every edge, } (u,v) \end{aligned}$$

Maximum Matching

- We saw how to solve problems like this when we studied flow networks

Suppose five children have gone to a dog shelter to find a new pet. As it happens, there are exactly five dogs up for adoption. Each child has a preference for certain breeds, whereas each dog has a preference for children of a certain temperament. Figure 26.6 shows a graph in which a child and a dog who like each other are connected by an edge.

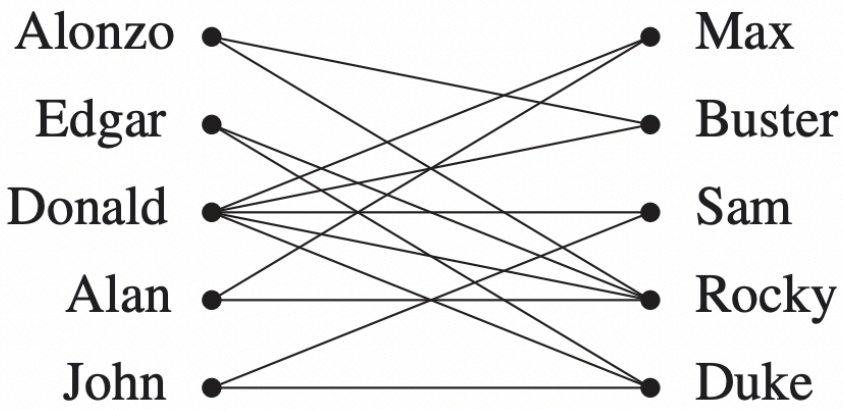


Figure 26.6: An edge connects a child and a dog who like each other.

Recall How We Solved It

Is there a pairing scheme in which all children and dogs will be happily matched? In other words, is there a maximum matching for the graph in Figure 26.6 equal to the number of children? In general, we can answer maximum matching questions by setting up a flow network, as done in Section 16.3, and then solving this problem using the LP given above. We create two additional vertices, a source s and a sink t. Then, we add a directed edge from the source to every child, and also an edge from every dog to the sink. Finally, we connect every child to a dog if they both like each other. We get the directed graph shown in Figure 26.7.

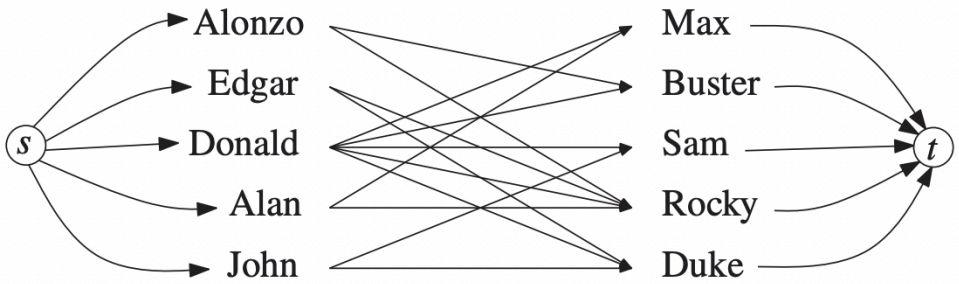


Figure 26.7: This flow network solves the bipartite matching problem.

Next, we assign a flow of 1 to each edge. Then there is a perfect matching in the original graph if and only if the maximum flow is equal to 5. Moreover, if we solve the flow network problem by converting it into an LP as we did earlier, all flows will be integers. This means that each edge can only have a flow of 0 or 1, so that we can interpret 0 as "do not pair them up" and 1 as "pair them up." Thus, we can solving the maximum matching problem using a linear program solver.

But Consider…

The maximum flow problem is to find the maximum flow size, that is, the maximum amount of flow out of the source that satisfies the above two rules. This can be expressed with a linear program in which the variables are the edge weights, $f(e)$, the objective function is the flow size, and the constraints are given by the two rules making up the definition of a valid flow:

maximize:	$\sum_{e \in E^+(s)} f(e)$	where s is the source
subject to:	$0 \leq f(e) \leq c(e)$	for all edges e
	$\sum_{e \in E^-(v)} f(e) = \sum_{e \in E^+(v)} f(e)$	for all vertices v except for the source and the sink

Here, $E^+(v)$ and $E^-(v)$ represent the set of incoming and outgoing edges of a vertex v respectively. Note that if the capacities of a flow network are integers, the simplex algorithm will produce a solution with integer edge weights.

Duality

To prove that Algorithm 26.5 does indeed provide the correct output, we must discuss a linear program related to the original problem called the ***dual***.

Suppose that our input LP has the standard form:

$$\begin{aligned} \text{maximize:} \quad & z = \sum_{j \in V} c_j x_j \\ \text{subject to:} \quad & \sum_{j \in V} a_{ij} x_j \leq b_i \text{ for } i \in C \\ & x_j \geq 0 \text{ for } j \in V \end{aligned}$$

It can be put into slack form as follows:

$$\begin{aligned} \text{maximize:} \quad & z = \sum_{j \in F} c_j x_j \\ \text{subject to:} \quad & x_i = b_i - \sum_{j \in F} a_{ij} x_j \text{ for } i \in B \\ & x_j \geq 0 \text{ for } j \in F \cup B \end{aligned}$$

Duality, continued

Therefore, $C = B$ and $V = F$. Note that this only holds because we did the obvious transformation between the two forms. There are many more slack forms equivalent to the original standard form, such as the slack forms produced at each iteration of Algorithm 26.5, whose indexing sets B and F do not directly correspond to C and V. Nevertheless, for this section, we assume that whenever we transform the standard form into slack form, we do the obvious transformation. So we can assume that $C = B$ and $V = F$.

Given this initial LP, the ***dual LP*** is a minimization problem which interchanges the roles of $\vec{b}$ and $\vec{c}$ and the roles of B and F. It also introduces new variables y_i:

$$\begin{aligned} &\text{minimize:} && z = \sum_{i \in B} b_i y_i \\ &\text{subject to:} && \sum_{i \in B} a_{ij} y_i \geq c_j \text{ for } j \in F \\ & && y_i \geq 0 \text{ for } i \in B \end{aligned}$$

When considering the original problem in relationship to its dual, we refer to the original as the ***primal*** LP. Note the symmetry between the primal and dual LPs written in matrix form:

$$\begin{array}{llll} & \text{maximize:} & z = \vec{c} \cdot \vec{x} & \\ \text{primal:} & \text{subject to:} & A\vec{x} \leq \vec{b} & \\ & & \vec{x} \geq \vec{0} & \end{array} \qquad \begin{array}{llll} & \text{minimize:} & z = \vec{b} \cdot \vec{y} & \\ \text{dual:} & \text{subject to:} & A^t\vec{y} \geq \vec{c} & \\ & & \vec{y} \geq \vec{0} & \end{array}$$

Duality Example

Example 26.5: *Below is a primal LP written in standard form and its dual.*

maximize:	$z = x_1 + 2x_2$	*minimize:*	$z = 3y_1 + 2y_2 + y_3$
subject to:	$-3x_1 + 2x_2 \leq 3$	*subject to:*	$-3y_1 + y_2 + y_3 \geq 1$
	$x_1 + x_2 \leq 2$		$2y_1 + y_2 - y_3 \geq 2$
	$x_1 - x_2 \leq 1$		$y_1, y_2, y_3 \geq 0$
	$x_1, x_2 \geq 0$		

Another Duality Example

Example 26.6: *Recall the primal LP representing the web server problem.*

$$\begin{aligned} \textit{maximize:} \quad & z = 1000x_1 + 2000x_2 \\ \textit{subject to:} \quad & 400x_1 + 1600x_2 \leq 36800 \\ & 2x_1 + x_2 \leq 44 \\ & 300x_1 + 500x_2 \leq 12200 \\ & x_1, x_2 \geq 0 \end{aligned}$$

The dual problem is:

$$\begin{aligned} \textit{minimize:} \quad & z = 36800y_1 + 44y_2 + 12200y_3 \\ \textit{subject to:} \quad & 400y_1 + 2y_2 + 300y_3 \geq 1000 \\ & 1600y_1 + y_2 + 500y_3 \geq 2000 \\ & y_1, y_2, y_3 \geq 0 \end{aligned}$$

So, How to Think About This

- Suppose a computer manufacturer claims it can meet the web company's needs, and will do so by offering servers that will outperform the two types of server company is considering
  - What the manufacturer is effectively offering is hits/min. It needs to provide the web company with a sufficient number of hits/min to get contract
  - BUT, hits/min are its resource, so minimizing the number of hits/min it offers (while still meeting requirements) effectively minimizes cost
- The manufacturer is given the resource available to the web company:
  - $36,800, 44 server shelves, 12200 W of power.
- Let us define new variables, whose units are (hits/min) per unit of resource
  - y_1 = (hits/min) per dollar
  - y_2 = (hits/min) per unit of shelving
  - y_3 = (hits/min) per Watt of power

So, How to Think About This

- Let us define new variables, who units are (hits/min) per unit of resource
  - y_1 = (hit/min) per dollar
  - y_2 = (hits/min) per unit of shelving
  - y_3 = (hits/min) per Watt of power
- So, in order to meet or exceed the standard server, we need

  $400y_1 + 2y_2 + 300y_3 \geq 1000$.
- In order to meet or exceed the cutting-edge server, we need

  $1600y_1 + y_2 + 500y_3 \geq 2000$.
- Note that each summand is giving hits/min
- Finally, we wish to minimize the hits/min required for this

  $36{,}800y_1 + 44y_2 + 12200y_3$
- It turns out that the minimum number of hits/min the manufacturer can get away with is equal to the maximum number of hits/min that the web company can achieve using the two types of server it had at its disposal!

Why The Dual?

Well, the concept is used to prove, among other things, that the simplex method works.

Theorem 26.8: *Suppose that on input LP* $(A, \vec{b}, \vec{c}, 0, F, B)$*, Algorithm 26.5 returns the LP* $(A', \vec{x}', \vec{c}', c_*, F', B')$*, then* $\vec{x}'$ *is optimal for the input LP.*

How is the dual used in the proof?

Lemma 26.7: *Let* $\vec{x}'$ *be a feasible solution to the primal LP* $(A, \vec{b}, \vec{c}, c_*, F, B)$ *and let* $\vec{y}'$ *be a feasible solution to the dual LP. If* $\vec{c} \cdot \vec{x}' = \vec{b} \cdot \vec{y}'$*, then* $\vec{x}'$ *and* $\vec{y}'$ *are optimal for their respective problems.*