The Linear Programming Problem and the Simplex Method

Now we want to study a problem, known as "linear programming," and one algorithm that solves this problem, known as the "simplex method."

A lot of real world problems, from mathematics, science, engineering, business, and economics, can be formulated as linear programming problems. In addition to its practical uses, our work in this section will give us an important case study in computational complexity in the last part of the course, when we study computational complexity and "P vs. NP."

Software packages that implement the simplex method have been well-developed for half a century, and are still in common use. When I taught MTH 3250 in the early 1980's, the text we used said that half the computer time being used on the planet was being devoted to performing the simplex method. Lists of the most important algorithms typically include the simplex method.

Earlier we touched briefly on numerical algorithms and decided not to pursue them further. Technically the simplex method is a numerical algorithm, subject to rounding error issues, but if it were executed with perfect accurate arithmetic computations, it would behave like a non-numerical algorithm.

The Linear Programming Problem (LP)

The linear programming problem, known henceforth as LP, is to find the optimal value of a function, subject to certain constraints on the arguments, where the function and the constraints are linear, namely of the form of a sum of terms consisting of a scalar times a variable, such as $3x_1 + 2x_2$.

An instance of LP has the standard form

---

max $c^T x$
subject to the constraints
$Ax = b$
$x \geq 0$

---

where standard matrix notation is used, n is the number of *decision variables*, m is the number of *equality constraints*, x and c are n by 1 matrices, A is m by n, b is m by 1, and for any matrix M, $M \geq 0$ means that all the components of M are non-negative. We also assume that $b \geq 0$.

Later we will see how all sorts of problems that don't have the standard form can be converted to an equivalent problem that fits this form. So, all our theory, and the algorithm we present for solving an LP, will be based on this form.

First Example of a LP Problem Suppose a farmer has 1200 acres of farm land to be planted in either corn or wheat. Suppose further that the farmer has 12000 units of water to be devoted to these crops, and 18000 monetary units with which to buy seed for the crops. Each acre of corn requires 8 units of water for the season, and each acre of wheat requires 15. Seed for an acre of corn costs 6 monetary units, while an acre of wheat costs on 30 units per acre. If the expected produce from an acre of corn will sell for 20 monetary units and an acre of wheat will produce 30 monetary units, how many acres of corn and wheat should the farmer decide to plant?

This word problem produces the LP instance

$$\max 20x + 30y$$

subject to the constraints

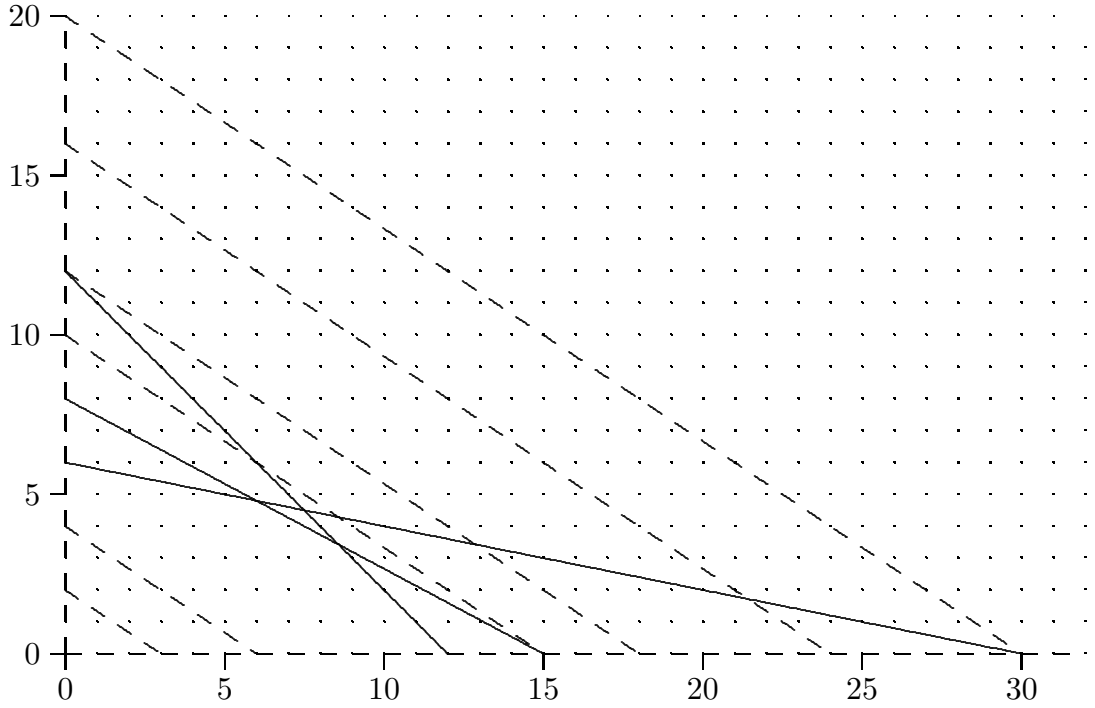
$$x + y \leq 1200$$

$$8x + 15y \leq 12000$$

$$6x + 30y \leq 18000$$

$$x, y \geq 0$$

We can graph the boundary lines for the constraints, and some level sets for the objective function (sets on which all points have the same function value) for this problem:



(in units of 100)

⇐ We can solve this instance of LP by finding the feasible region—the region consisting of points (x, y) that satisfy all the constraints—and then finding the level set which crosses

the feasible set and has the largest objective function value. We can do this graphically, or by solving pairs of linear equations to find candidates for the winning point.

Brief Theory for Linear Programming

Now we will take a brief look at some mathematical theory for the linear programming problem, intended to provide intuition for the simplex method. Basically, we want to be able to view a general LP in the same way we handled the graphical picture in our previous example.

Definitions

Define $F = \{x | Ax = b \text{ and } x \geq 0\}$. The set F is known as the "feasible set," and we sometimes say "x is feasible" instead of $x \in F$.

A set S is *convex* if for every u and v in S, and for every $\lambda \in [0, 1]$, $(1 - \lambda)u + \lambda v \in S$.

$\Leftarrow$ Note that any point on the line passing through u and v has the form $u + \lambda(v - u)$ for some number λ. Figure out what values of λ produce a point that lies between u and v. Do some algebra to rewrite this quantity with u and v appearing just once. Now the definition should make sense—draw some 2D sets that are convex and some that are not convex.

We will refer to the sets of points on the line between two points in space as a "segment" (short for "line segment"). With this terminology, a set is convex if for every pair of points in it, the segment connecting them lies entirely in the set.

Theorem F is convex.

$\Leftarrow$ Prove this theorem by using algebraic properties of the equations and inequalities that specify the feasible set.

Theorem Suppose that x^* is a solution to an instance of LP, and that x^* is a *strict optimum* point, meaning that for all $x \in F$, if $x \neq x^*$ then $c^T x < c^T x^*$. Then x^* is an *extreme point* of F, which is defined to mean that x^* does not lie in the interior (i.e., is not one of the two endpoints) of any segment that is a subset of F.

Proof: Suppose to the contrary that x^* is a strict maximum of an instance of LP, and that there are points u and v in F with $u \neq v$ and x^* in the interior of the segment from u to v. So, for some $\lambda \in (0, 1)$, $x^* = (1 - \lambda)u + \lambda v$. But then simple matrix algebra, and the fact that x^* is a strict maximum, shows that

$$c^T x^* = (1 - \lambda)c^T u + \lambda c^T v < (1 - \lambda)c^T x^* + \lambda c^T x^* = (1 - \lambda + \lambda)c^T x^* = c^T x^*,$$

which is a contradiction.

$\Leftarrow$ Verify all the details in this proof.

Theorem Suppose that some $x \in F$ has at least $m+1$ positive components. Then x is not an extreme point of F, and therefore it is not a strict local maximum of the LP instance.

Proof: For notational convenience, assume that x has exactly $m+1$ positive components, and that they are the first $m+1$. The proof would be similar but harder to describe in the general case.

Denote column k of A by A_k. By basic matrix algebra, it is easy to see that the system of equations

$$A \begin{bmatrix} y_1 \\ \vdots \\ y_{m+1} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = 0$$

has a non-zero solution. In other words, there is a vector y such that y_1, y_2, ..., y_{m+1} are not all zero, y_{m+2}, ..., y_n are all zero, and $Ay = 0$.

Now, it is easy to see that for all small enough numbers μ, both positive and negative, the points $x + \mu y$ are feasible. Thus, x lies on an interval of positive length that is a subset of F, so x is not an extreme point.

$\Leftarrow$ Fill in the many missing details in this argument.

Theorem Suppose we pick m columns of A and denote them as "basic," and want to solve $Ax = b$ for the corresponding m components of x while setting the other $n - m$ components of x to 0. Suppose that these columns row reduce to the identity, making the solution possible, and that $x \geq 0$. Then the point x is an extreme point of F, and hence might be a strict local maximum of the LP.

Proof: Again, for notational convenience, assume that the first m columns of A are the ones we are considering "basic." Partition $A = [B\ N]$, where B is the non-singular m by m matrix composed of the first m columns of A, and N is the remaining $n - m$ columns.

When we do row operations to row reduce B to the identity, the matrix is transformed to

$$[I \quad B^{-1}N \quad | \quad B^{-1}b \quad],$$

and $x = \begin{bmatrix} B^{-1}b \\ 0 \\ \vdots \\ 0 \end{bmatrix}$.

Now, suppose that x is not extreme, implying that there are points u and v in F such that $x = (1 - \lambda)u + \lambda v$, for some $\lambda \in (0, 1)$, with $u \neq v$. It is easy to see that u and v must have their components equal to 0 anywhere that x does, including all those below row m, and then it is easy to see, since the square, non-singular system of equations $Bz = b$ has a unique solution, that we must in fact have $u = v = x$.

$\Leftarrow$ Go through all the steps of this outline and fill in the details.

A Brute Force Method for Solving LP

The preceding theory was intended to give us an intuitive feel for the simplex method. We can now use this theory to develop a brute force method for solving an instance of LP that will lead naturally to the simplex method.

The brute force method is based on this idea: if there is a value for x that beats all other (a strict maximum point—not tied with any other feasible point) then that point is an extreme point of the feasible set, and extreme points can be obtained by picking m columns to be basic, setting the variables for the non-basic columns to be 0, and solving for the basic variables.

More precisely, here is the brute force method: for each of the $\binom{n}{m}$ choices of subsets of m columns of A, try to use them as basic columns and solve for x, setting the other components of x to 0. For some choices of m columns, it may be impossible to do this. For choices for which it is possible, we first check to see if all the components of x turn out to be non-negative. If they are not, then the point is not feasible. If they are, then the point is feasible, and we compute $c^T x$ and note that value. When we have examined all the possibilities, the one (or several, perhaps, in case of a tie) with the largest function value is the optimum point.

⇐ Solve the "farm" problem by the brute force method, using the `MatrixManipulator` application in the folder `Simplex`. Note that I'll use this program today in class, but as soon as I finish the new version of the tool, we'll use it.

The Simplex Method for Solving LP

We can now describe, and more importantly, understand, the simplex method. The simplex method simply maintains a set of basic columns of A at all times (later we will discuss how to get the initial set), requiring the corresponding x to be non-negative, and moves from one basic set to another by replacing one column with a better one.

Now for the details. First, we add an *objective function row* at the top, and a column at the left, corresponding to a made-up variable z, where we say that $z = c^T x$. To put this equation, which defines z as the objective function value at x, in the same form as the $Ax = b$ constraints, we simply transform it to $z - c^T x = 0$.

Now the matrix, which is traditionally known as a *tableau*, looks like this:

	z	x_B^T	x_N^T	
	1	$-c_B^T$	$-c_N^T$	0
x_B	$\begin{matrix}0\\0\\\vdots\\0\end{matrix}$	B	N	b

where for notational convenience we pretend that it is the first m columns that have been selected as basic, and denote those columns of A by B, and use x_B and x_N to represent, respectively, the variables corresponding to the basic and non-basic columns. Similarly, c_B and c_N are the parts of c that correspond to the basic and non-basic columns.

> When we actually perform the simplex method, the m basic columns will be scattered throughout A, and we don't actually need to switch columns to put them at the front, though we could.
>
> Also note that the x_B^T and x_N^T blocks are just there as column labels to indicate which variables correspond to which columns.
>
> The column for the z variable never changes, and is only there to remind us of what the objection function row means as we develop the algorithm.

Now, to derive the simplex method details, we simply do some block matrix operations, using the powerful fact that if you have a matrix partitioned into blocks, then any operations that make sense with individual numbers in the matrix also work with blocks. So, we perform block row operations to solve for x_B in terms of other quantities.

First, we multiply the "second row" of the partitioned matrix by B^{-1} (which of course only works if this matrix is non-singular), obtaining the equivalent set of equations

	z	x_B^T	x_N^T	
		$-c_B^T$	$-c_N^T$	0
x_B	$\begin{matrix}0\\0\\\vdots\\0\end{matrix}$	I	$B^{-1}N$	$B^{-1}b$

Then we add the correct multiple of "row 2" to "row 1" to zero out "row 2" in "column 1," namely c_B^T, obtaining the tableau

	z	x_B^T	x_N^T	
	1	0^T	$c_B^T B^{-1}N - c_N^T$	$c_B^T B^{-1}b$
x_B	$\begin{matrix}0\\0\\\vdots\\0\end{matrix}$	I	$B^{-1}N$	$B^{-1}b$

Now, the "second row" says that

$$Ix_B + B^{-1}Nx_n = B^{-1}b,$$

so if we set $x_N = 0$, these equations say that $x_B = B^{-1}b$.

The "first row" says that

$$z + 0^T x_B + (c_B^T B^{-1} N - c_N^T)0 = c_B^T B^{-1} b,$$

or

$$z = c_B^T B^{-1} b.$$

Thus, the rightmost column of the tableau gives the value of the objective function corresponding to the current choice of basic variables, and the elements below that in the rightmost column give the values of the basic variables.

So, if we were to follow a brute force approach, we would simply keep changing our minds about which m columns should be basic and doing row operations to make those columns into the corresponding columns of the identity (though rearranged), and monitoring the upper right corner value and the right-hand-side values. For each choice of basic columns, if B is non-singular, then we check to see if $B^{-1}b$ has all non-negative components, and if it does, then we note the upper right corner value as a possible optimal value. After trying all possible sets of m columns out of n, we would have the winner.

The simplex method is more clever than this, of course. We will now develop the remaining details of how it works.

The idea is to let one non-basic column, say column j, sneak into being basic. In other words, we examine what would happen if we were to let x_j increase to some positive value, say α. If we denote the value in column j of the row matrix

$$c_B^T B^{-1} N - c_N^T$$

(which is known as the "reduced costs") by γ, then the first row says

$$z + \gamma\alpha = c_B^T B^{-1} b,$$

so the new objective function value would be the previous one ($c_B^T B^{-1} b$) minus $\gamma\alpha$. Thus, since α is positive, the objective function will be improved if γ is negative.

So, if we see a negative number in the objective function row, then we think (though we would be wrong in some cases, as we will see) that we can improve the objective function value by letting the non-basic variable corresponding to a negative reduced cost increase from 0 to some positive value.

Of course, why not let it increase a whole bunch and really improve? The catch, of course, is that as we let the value of x_j increase, the values of some of the x_B variables may decrease, and we can't let any of them go negative.

To be more precise, denote column j of the constraints part of the tableau by u, and note that the equations now say

$$x_B + \alpha u = B^{-1} b,$$

or

$$x_B = B^{-1}b - \alpha u.$$

So, for any u_i that is positive, increasing x_j to α reduces the value of the ith basic variable by α times that positive amount. Since we must maintain the $x \geq 0$ constraint of the LP, the amount that α can increase is restricted. Precisely, if we denote $B^{-1}b$ by r, then for each i such that $u_i > 0$, we must have

$$r_i - \alpha u_i \geq 0,$$

or

$$\alpha \leq \frac{r_i}{u_i}.$$

So, in case you missed it, that's the simplex method: given a good set of basic variables, do row operations to transform the corresponding columns of A to an m by m identity matrix, and then examine the reduced costs. If any of them are negative, pick one that is negative to enter, find the minimum ratio of right-hand-side elements to positive elements of its column, and increase the non-basic variable by that amount. In so doing, the previously basic variable for that row will be forced to 0, and can safely be considered non-basic, replaced by the newcomer. When there are no negative reduced costs, we have found the optimal point.

⇐ Now we can apply the brute force approach and the simplex method to the farm LP instance.

⇐ Analyze how the simplex method can terminate. In particular, what if we have a column with a negative reduced cost where there are no positive values below in the column?

Getting a Good Initial Choice of Basic Columns

Our one example is misleading in the sense that there is a nice so-called slack variable for each constraint, so it is easy to find an initial choice of columns to be basic. Consider the general problem of finding an initial choice B of columns to be basic. We want two things of these columns. First, B must be non-singular, so that we can perform row operations to transform it to the identity matrix. Second, $B^{-1}b$ must be non-negative, because that vector holds the values of the basic variables.

Note that if we can find a choice of columns with these properties, then we will also have found a feasible point, namely the extreme point obtained from B. It turns out that the problem of determining whether there are any feasible points is just as hard, in an efficiency analysis sense, as solving any LP.

The technique we will use to find a good initial choice of basics is known as "Phase 1." Then the rest of the simplex algorithm is known as "Phase 2." Both phases use the same algorithm, namely moving from extreme point to extreme point, but Phase 1 uses a different objective function than the original one, and then Phase 2 proceeds with the actual objective function.

Once we have modeled the original problem in the $Ax = b$ form (we will see later how to handle inequalities and unrestricted variables), we first look for columns of A that are already columns of the identity. If such a column has its 1 in row i, we check to see if b_i is non-negative. If it is, then we might as well use that column as basic. This is done simply to save time. The upcoming technique could be used without this shortcut.

The idea is simple: for each row k that does not already have an obvious basic variable, we first multiply the row by -1, if necessary, to make sure that b_k is non-negative, and then we make up a new variable, known as an artificial variable, say named a_k, and simply add a_k to the left hand side of the equation for that row. This guarantees that the column for a_k is a column of the identity, with a 1 in row k, and a positive right-hand side value, so a_k is a good basic variable for row k.

After doing this as necessary until we have all m columns of the m by m identity matrix identified as basic columns, with all the right-hand side values non-negative, we have a very nice starting point for the simplex method. Unfortunately, this problem is not our original problem, so all this appears to make little sense. The trick is to use as the objective function the sum of the artificial variables.

Using this special Phase 1 objective function, we perform the simplex method until we have to stop, either because the problem is unbounded, or because we find an optimal extreme point.

⇐ Can a Phase 1 problem ever be unbounded? Why not?

So, Phase 1 will always terminate with an optimal extreme point. If the original problem has a feasible point, then the Phase 1 problem will have an optimal point with no artificial variables in the set of basic variables, with an objective function value of 0. If the original problem is infeasible, then the Phase 1 problem will terminate with one or more artificial variables still basic.

If the Phase 1 simplex method produces an optimal value of the objective function that is positive, then we conclude that the original problem is infeasible. Otherwise, we make sure that no artificial variables are still basic by perhaps pivoting some more. Then, we replace the artificial objective function by the real one, and happily proceed with the simplex method.

⇐ Think about how an artificial variable could be basic and still have the Phase 1 objective function be 0. Variables in this situation indicate a case where two or more extreme points are tied for best, in which case we can pivot to swap them however we like.

When doing the Phase1-Phase 2 simplex method, be sure before starting either phase that you "price out" the objective function row. It is clear from our theory that the upper-right corner is the current objective function value only if the row operations have been done to make zeros in the objective function row in the basic columns. When we add in an artificial variable with a nice identity column, and a 1 in the objective function row, it is easy to forget that we still have to pivot on the 1 in the identity column in order to transform the 1 in the objective function row to a 0. Then, once Phase 1 concludes, we

swap in $-c^T$ in the objective function row, and we again need to remember to pivot on the 1's in the various basic columns to zero out those same columns in the objective function row.

Of course, for Phase 2 we either delete the artificial variable columns, or ignore them. In particular, if a negative reduced cost appears in an artificial variable column, we must not let the artificial variable become basic.

With this Phase 1-Phase 2 technique we now have an actual algorithm that can solve any instance of LP.

The Transportation Problem

As a second example of an LP, known as the "transportation problem," consider the following situation. Suppose we have some factories, each of which produces some number of units of some product, and some stores, each of which wants to receive some number of units of the product, where the number of units produced is equal to the number of units desired. Further, suppose there is a per-unit cost to ship a unit from any factory to any store. The optimization problem is to decide how many units to ship from each factory to each store.

Here is all the data for an instance of the transportation problem, where the last column is the number of units produced by each factory, the last row is the number of units to be shipped to each store, and row i, column j of the 3 by 5 array is the cost to send a unit from factory i to store j:

	1	2	3	4	5	
1	3	7	11	4	2	10
2	5	9	4	2	8	15
3	6	1	9	4	7	12
	8	6	10	7	6	

⇐ If we let x_{ij} be the number of units sent from factory i to store j, we can formulate the transportation problem as an LP, create the simplex method tableau, and solve the problem. Note that since the total units produced at the factories is the same as the total number of units desired at the stores, we can use equality constraints, but then we'll need to do the artificial variables technique to find an initial feasible basis.

Degeneracy and Cycling

One issue muddles up the clarity of the simplex method. When a basic variable ends up having a value of 0—referred to as a *degenerate* case—it is possible to do a pivot step of the simplex method without actually moving in space—the set of basic variables changes, but we don't move to a new extreme point and we don't improve the objective function value. It is possible in this situation for *cycling* to occur, where, depending on how the specific implementation of the simplex method chooses which non-basic variable with a negative reduced cost to become basic, and how ties are broken between rows with the same minimum ratio, the simplex method can cycle through the same pivot steps, never actually moving, forever. It is easy to make choices, however, that prevent this from happening.

As a practical matter for us, note that if an artificial variable is basic, but with a value of 0, to keep things simple we should replace it in the set of basic variables by a non-artificial variable, even though the objective function stays at 0.

Some Modeling Techniques

At first glance, the form of LP seems to rule out lots of interesting things, such as inequality constraints, unrestricted variables, and minimization, but all of these can easily be handled by some modeling tricks.

First, if your objective is to minimize $c^T x$, simply maximize $-c^T x$ instead, but always remember that the upper-right corner value is the opposite of your true objective function value.

One trivial but important idea is that you must sometimes multiply both sides of a constraint—equality or inequality form—by -1 so that the right hand side value is positive, as required by the simplex method.

Inequality constraints can easily be converted to equality constraints by noting that $a^T x \leq \beta$ is equivalent to $a^T x + \alpha = \beta$ and $\alpha \geq 0$. Since the simplex method keeps all variables non-negative, we can simply introduce α as a new variable and replace the inequality form by the equality form. This kind of new variable, which we already used to solve our first example, is known as a "slack variable."

$\Leftarrow$ Figure out how to similarly handle $\geq$ type constraints by using what is known as a "surplus variable."

Finally, if a variable x_k in the problem is unrestricted, meaning we want to allow for the possibility of it being negative, we simply replace it everywhere in the problem by using the substitution $x_k = x_k^+ - x_k^-$, where x_k is now no longer a variable in our problem, and we have two new variables x_k^+ and x_k^-, both of which are required to be non-negative. This technique works because any number can be represented as the difference of two positive numbers.

As part of this last idea, we can put a term like $|x_3|$ in either the objective function or the constraints as $x_3^+ + x_3^-$, if x_3 is unrestricted.

⇐ Note that in fact there are infinitely many ways to do this, namely to express a given number—positive or negative—as the difference of two other numbers, both of which are nonnegative. This is not a problem because the simplex method forces one of the numbers to be 0.

Example of Formulating and Solving an LP

Suppose we are writing a 2D game engine and want to compute whether, and if so, at what time, two moving rectangles will first touch each other.

After some thought, we figure out that two moving boxes (by "moving" we mean straight-line motion without rotation, because for anything more complex, the problem would be non-linear) are equivalent to one stationary box in a simple position and orientation, with the other box with arbitrary position, orientation, and velocity.

So, assume that the stationary box has corners at $(0, 0)$, $(w, 0)$, (w, h), and $(0, h)$, where w and h are arbitrary positive numbers.

And, assume that the moving box has corners at (c_x, c_y), (c_x+r_x, c_y+r_y), (c_x+u_x, c_y+u_y), and $(c_x + r_x + u_x, c_y + r_y + u_y)$, where all these quantities are arbitary real numbers, but that the vectors (r_x, r_y) and (u_x, u_y) are perpendicular (so $r_x u_x + r_y u_y = 0$). Finally, assume that the box has velocity vector (d_x, d_y).

⇐ This all sounds pretty horrible, but a sketch will make it quite clear.

Now we need to model the motion and the collision. We note that the boxes are solid, and note that after the box has moved for time $\lambda \geq 0$, an arbitrary point (determined by some $\alpha \in [0, 1]$ and $\beta \in [0, 1]$) in the moving box will be at the position

$$(c_x + \alpha r_x + \beta u_x + \lambda d_x, c_y + \alpha r_y + \beta u_y + \lambda d_y).$$

Finally, the constraints on the problem are given by saying that this arbitrary point, after moving for time λ, will touch the stationary box (which means the point will be on the border or inside the stationary box). And, the objective function is simply to minimize λ subject to these constraints, thus finding the first time at which the boxes touch.

We can now take some specific values for the various parameters that define the two boxes and manually create an LP instance. Then we can use the simplex method to solve the instance, thus determining whether/when the two boxes will collide.

Take the stationary box to have $w = 3$ and $h = 5$, and model the moving box by $c = \begin{bmatrix} 13 \\ 11 \end{bmatrix}$, $r = \begin{bmatrix} -1 \\ 3 \end{bmatrix}$, $u = \begin{bmatrix} -6 \\ -2 \end{bmatrix}$, and $d = \begin{bmatrix} -2 \\ -1 \end{bmatrix}$.

⇐ Formulate and solve a problem of this type for some specific numbers.

Another Example of Formulating and Solving an LP

Consider the maximum-flow problem: given a directed graph with a maximum flow capacity on each edge, determine the maximum flow from a designated source vertex to a designated sink vertex, where the source vertex has no edges going into it and the sink vertex has no edges going out of it. We assume that for any flow the amount flowing into any vertex is equal to the amount flowing out of that vertex.

⇐ Create a random small example of a maximum-flow problem, figure out how to model this problem as an LP, and solve it.

Instructions for the next three exercises: For each of the following exercises, you will be asked to "formulate and/or solve" a given LP. You should write down very carefully, using algebraic notation, the LP you are solving, enter the coefficients and row and column labels into a data file, and use `ManualSimplex` in the `Simplex` folder at the course web site to perform the simplex method on the given tableau. As you do this, write down, to turn in along with your algebraic formulation of the LP, the basic variables, their values, and the value of the objective function value at each step of the simplex method. Actually, you only need to write down the values of the decision variables—the original variables in the problem, not the slack/surplus/artificial variables you add, just to sanity check your steps. If you need to do Phase 1, clearly indicate the objective function you are using for Phase 1 and for Phase 2. On Test 2 you will be given tableaux and asked questions about the simplex method related to those tableaux, so make sure that you understand how the minimum ratio and pivot commands are operating.

Exercise 11 [4 points]

a. Formulate and solve this LP:

$$\max\ x_1 + 2x_2 + 3x_3$$

subject to the constraints

$$-3x_1 + 15x_2 - 3x_3 \geq 3$$

$$6x_1 + 3x_2 + 6x_3 \leq 60$$

$$-6x_1 + 6x_2 + 3x_3 \leq 21$$

$$9x_1 + 5x_2 - x_3 \geq 21$$

$$-3x_1 + 5x_2 + 2x_3 \geq 3$$

$$6x_1 + 8x_2 - 4x_3 \leq 30$$

$$8x_2 - 4x_3 \leq 12$$

$$3x_1 + 3x_3 \geq 12$$

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

b. Formulate and solve the above LP, but with this constraint added:

$$2x_3 \leq 1$$

c. Formulate and solve the LP for part a, but with the first four constraints removed.

Exercise 12 [3 points]

Formulate and solve the LP produced by the transportation problem with this data:

	1	2	3	4	5	6	
1	12	17	13	19	20	15	24
2	10	8	12	14	13	6	18
3	19	15	21	11	14	20	22
4	17	14	17	10	16	18	16
	13	12	10	14	15	16	