

THE SIMPLEX ALGORITHM (in one page)

The following steps form the fundamentals of the Simplex Algorithm for solving linear programs.

1) Set up an "initial tableau."

2) Is the trial solution represented by the initial tableau optimal?

If all coefficients in Row 0 are ≥ 0 , then the solution is optimal and you may STOP.

If any coefficients in Row 0 of the tableau are < 0 , the solution represented by this tableau is not the optimal solution and the set of basic variables must be changed.

3) Select a new basic variable.

Select as a new basic variable the variable with the most negative coefficient in Row 0. The column for this variable is the "pivot column."

4) Select a new non-basic variable.

Divide the RHS entry of each row by the coefficient in the column of the new basic variable.

The new non-basic variable is the old basic variable associated with the row having the minimum ratio. This row is called the "pivot row."

5) Update the tableau.

Form a new tableau that has zero for each entry in the pivot column, except for the coefficient in the pivot row. The entry in both the pivot row and pivot column must be equal to one. Use linear algebra and Gaussian elimination techniques to do this.

Replace the old basic variable with the new basic variable in the "Basic Variable" column.

This new tableau represents a new corner-point solution.

6) Is this new solution optimal?

If all coefficients in Row 0 of the new tableau are ≥ 0 , then STOP; the solution is optimal.

If any coefficients in Row 0 of the tableau are < 0 , the solution represented by this tableau is not the optimal solution and the set of basic variables must be changed. GO TO STEP 3.

Reading the final tableau.

The following tableau represents an optimal solution and final tableau for the problem:

$$\begin{array}{ll} \text{MAX} & 2X_1 + X_2 \\ \text{Subject To:} & 1) X_1 + X_2 \leq 4 \\ & 2) X_2 \leq 3 \end{array}$$

Basic Variable	Eqn. No.	Coefficient of Variable					RHS
		Z	X1	X2	S1	S2	
Z	0	1	0	1	2	0	8
X1	1	0	1	1	1	0	4
S2	2	0	0	1	0	1	3

The values of each basic variable is given by the RHS column. The value of each non-basic variable is zero. $Z = 8$, $X_1 = 4$, $S_2 = 3$, $X_2 = 0$, $S_1 = 0$.

The value of the Lagrange multiplier (shadow price) for each constraint is given by the coefficient in Row 0 under the associated slack variable (S1 or S2). $\lambda_1 = 2$; $\lambda_2 = 0$.

The Reduced Cost of each coefficient in the objective function is given by the coefficient in Row 0 under the associated decision variable (X1 or X2).

Can you obtain these results?

Some Simplex Details

The basic simplex method described earlier applies only to a relatively narrow set of linear programming problems. In particular, the problem must be a MINIMIZATION problem and all constraints must be of the form $\sum a_{ij} X_i \leq b_j$, i.e., all constraints must be less than or equal to constraints. There are also some ambiguities in the application of the simplex method. This lecture will try to clear up these problems.

Maximization

The simplex method minimizes the value of an objective function. But note:

$$\text{Max } z = \text{Min } (-z)$$

$$\text{Min } z = \text{Max } (-z).$$

So we can always make a maximization problem into a minimization problem, or vice versa.

Ties for New Basic Variable

If two or more variables have the same negative coefficient in Row 0 of the tableau, and that coefficient is the most negative coefficient in Row 0, then select any variable from those with the tie. The choice is arbitrary.

Ties for New Non-Basic Variable

What if 2 or more variables tie on the minimum ratio test for becoming the new non-basic variable?

Break the tie arbitrarily.

Possibility of cycling problems (discussed in book). Very rare.

Unbound Objective Function (Max $z \rightarrow \infty$ or Min $z \rightarrow \infty$)

An unbound solution occurs if the new non-basic variable has a RHS/pivot ratio = ∞ , i.e., if $\text{Min}(\text{RHS}/\text{pivot}) = \infty$. For this case that value of the new basic variable is not limited by any constraint and becomes infinite.

Multiple Optima

Where the optima fall on two or more corner-points, the optima will also include the edges of the feasible region between these points.

These cases will appear only when, after the STOPPING criterion has been reached, at least one non-basic variable has a coefficient of zero in Row 0. If this variable is increased (from zero) it will not affect the objective function. The other corner optimal corner point solutions are found by performing extra tableau iterations.

Equality Constraints

Equality constraints are of the form:

$$\sum a_{ij} X_i = b_j, b_j \neq 0.$$

The problem with equality constraints is that the initial solution, $X = 0$, is no longer feasible.

To fix this, we add a new type of slack variable, called an **artificial** variable. This type of slack variable acts as a regular slack variable in the simplex method. The new equality constraint becomes:

$$\sum a_{ij} X_i + R_j = b_j$$

We also add a huge penalty to the objective function if $R_j > 0$. The new objective function becomes:

$$\text{Min } z = \sum c_i X_i + M R_j.$$

M is a **HUGE** number compared to the other coefficients c_i .

These changes force the simplex method to quickly move away from the initially infeasible initial solution to a real feasible corner point.

This approach is called the **BIG M** method.

 \geq Inequality Constraints

\geq inequality constraints are of the form:

$$\sum a_{ij} X_i \geq b_j,$$

or, with a slack variable,

$$\sum a_{ij} X_i - S_j = b_j.$$

The same problem arises with \geq inequality constraints as with equality constraints, i.e., the initial tableau does not represent a feasible solution. Unless $b_j \leq 0$, X cannot be equal to zero and the initial trial solution is infeasible.

We solve this problem again using the Big M method.

First, an artificial variable is added to the constraint:

$$\sum a_{ij} X_i - S_j + R_j = b_j.$$

Second, a large penalty is given for having $R_j > 0$ by adding a HUGE coefficient for R_j in the objective function:

$$\text{Min } z = \sum c_i X_i + M R_j, \text{ where } M \text{ is very large.}$$

This quickly drives $R_j \rightarrow 0$ after the initial tableau and points the simplex method towards the feasible region.

Essentially, the BIG M method allows infeasible solutions, but only at an overwhelming penalty.

Sometimes constraints have the form:

$$\sum a_{ij} X_i \leq b_j, \text{ where } b_j < 0.$$

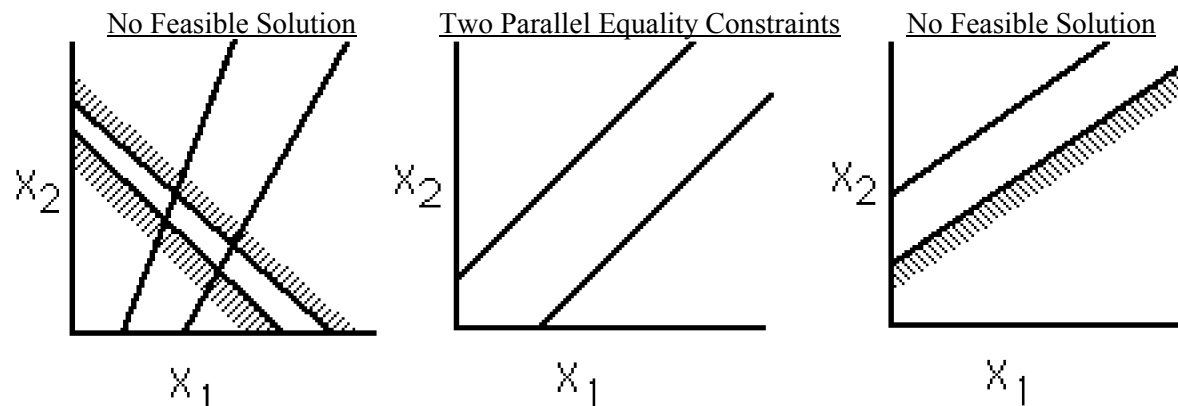
This can cause the initial tableau solution (all $X = 0$) to be infeasible. (This problem persists if we multiply both sides by negative one and reverse the inequality and becomes similar to solving a \geq inequality constraint.) We solve this again using the Big M method.

$$\sum a_{ij} X_i + S_j - R_j = b_j \quad \text{or} \quad -\sum a_{ij} X_i - S_j + R_j = \text{ABS}(b_j)$$

$$\text{Min } z = \sum c_i X_i + M R_j, \text{ where } M \text{ is very large.}$$

No Feasible Solution

This case can only occur if there are equality constraints or \geq inequality constraints in addition to regular \leq constraints. Therefore, artificial variables must appear in the formulation.



This case is discovered if the solution to the problem has at least one artificial variable not equal to zero.

Allowing Negative Values for a Decision Variable

How could we let a decision variable X_i have any value, including negative?

Must divide X_i into two new decision variables: X_i^+ and X_i^- . X_i^+ will represent X_i if it has a positive value and X_i^- will represent negative values of X_i .

We then replace X_i in all constraints and the objective function with:

$$X_i = X_i^+ - X_i^-.$$

We can similarly add a lower bound to the negative value of X_i by adding another constraint, where L_i is the absolute value of the lower negative bound:

$$X_i^- \leq L_i.$$

This ensures that $X_i \geq -L_i$.

Implementing the Big-M Method

We've suggested this method a lot. What does it do to the simplex method. A simple example:

$$\begin{array}{ll} \text{Min} & 5 X_1 + 2 X_2 \\ \text{Subject to:} & X_1 \geq 1 \\ & 2 X_1 + X_2 = 4 \\ & 3 X_1 + 4 X_2 \leq 30 \end{array}$$

Change this to:

$$\begin{array}{ll} \text{Min} & 5 X_1 + 2 X_2 + M R_1 + M R_2 \\ \text{Subject to:} & X_1 - S_1 + R_1 = 1 \\ & 2 X_1 + X_2 + R_2 = 4 \\ & 3 X_1 + 4 X_2 + S_3 = 30 \end{array}$$

Note that the 3rd constraint, a \leq constraint, does not require an artificial variable.

The first initial tableau:

Basic Variable	Equation No.	Coefficients of Variables							RHS
		z	X ₁	X ₂	S ₁	S ₃	R ₁	R ₂	
	0	1	5	2	0	0	M	M	0
	1	0	1	0	-1	0	1	0	1
	2	0	2	1	0	0	0	1	4
	3	0	3	4	0	1	0	0	30

Note that the "artificial variables" R₁ and R₂ are not yet basic variables, however. (We want to start at the origin.) They need to have a coefficient of zero in Row 0. We have to fix this before we begin the simplex method, using linear algebra operations (Gaussian elimination). Multiplying Row 1 by M and subtracting it from Row 0 makes R₁ basic, and multiplying Row 2 by M and subtracting this from Row 0 makes R₂ basic. This becomes the true initial tableau. This results in:

Basic Variable	Equation No.	Coefficients of Variables							RHS
		z	X ₁	X ₂	S ₁	S ₃	R ₁	R ₂	
z	0	1	5-3M	2-M	M	0	0	0	-5M
R ₁	1	0	1	0	-1	0	1	0	1
R ₂	2	0	2	1	0	0	0	1	4
S ₃	3	0	3	4	0	1	0	0	30

This tableau represents the corner-point "solution" at the origin ($X_1 = X_2 = 0$). This pseudo-solution is penalized by its poor performance of 5M.

Ah! That's better. Now we start the simplex method using this as our initial tableau.

To complete the example, following the standard Simplex algorithm,

Step 2. This solution is not optimal, several coefficients in Row 0 are negative.

Step 3. The X_1 column has the most negative value, making this the "pivot column" and X_1 the new basic variable.

Step 4. Eqn./Row 1 has the minimum ratio of RHS/coefficient in the X_1 column ($=1/1$), making this the "pivot row," and R_1 the new non-basic variable.

Step 5. Updating the tableau to make X_1 basic, the new tableau becomes:

Basic Variable	Equation No.	z	Coefficients of Variables				R_1	R_2	RHS
			X_1	X_2	S_1	S_3			
z	0	1	0	$2-M$	$5-2M$	0	$-5+3M$	0	$-5-2M$
X_1	1	0	1	0	-1	0	1	0	1
R_2	2	0	0	1	2	0	-2	1	2
S_3	3	0	0	4	3	1	-3	0	27

Step 6. This tableau is not optimal. Continue the algorithm with Step 3.

Step 3. S_1 is the new basic variable/pivot column.

Step 4. Row 2 is the new non-basic variable. $(2/2)=1$ (Row 1 is excluded because of the negative coef.)

Step 5. The new tableau is:

Basic Variable	Equation No.	z	Coefficients of Variables				R_1	R_2	RHS
			X_1	X_2	S_1	S_3			
z	0	1	0	$-1/2$	0	0	M	$-2.5+M$	-10
X_1	1	0	1	$1/2$	0	0	0	$1/2$	2
S_1	2	0	0	$1/2$	1	0	-1	$1/2$	1
S_3	3	0	0	$5/2$	0	1	0	$-3/2$	24

Step 6. This tableau is not optimal. Continue the algorithm with Step 3.

Step 3. X_2 is the new basic variable/pivot column.

Step 4. Row 2 is the new non-basic variable. $(1/0.5)=2$

Step 5. The new tableau is:

Basic Variable	Equation No.	z	Coefficients of Variables				R_1	R_2	RHS
			X_1	X_2	S_1	S_3			
z	0	1	0	0	1	0	$-1+M$	$-2+M$	-9
X_1	1	0	1	0	-1	0	1	0	1
X_2	2	0	0	1	2	0	-2	1	2
S_3	3	0	0	0	-5	1	-5	-4	19

Step 6. This tableau is optimal, with the solution $X_1 = 1$, $X_2 = 2$, $z = 9$.

Sensitivity Analysis in Linear Programming

Sensitivity Analysis: How would the solution change if parameter or coefficient values are varied?

Often for real problems parameter values are not known with certainty. Thus sensitivity analysis can tell us the relative importance of different sources of uncertainty.

Sensitivity analysis is the systematic variation of parameter and input data values in a model to assess the affect of uncertainties or variation in these variables on model results and designs based on model output.

Sensitivity analysis helps answer such questions as:

- 1) What is the effect if parameter c_i , b_j , or a_{ij} is off by some error ϵ ?
- 2) If input data values are accurate within some error ϵ , what are possible effects on model results?
- 3) Is further research into the value of certain parameters and input data warranted?

Since many models have large numbers of parameters and input data, conducting a complete sensitivity analysis on all parameters and data, and all possible combinations of co-varied errors, is impossible. This implies that sensitivity analysis must be somewhat judicious, concentrating on those parameters and inputs that seem most important and prone to error.

Brute Force Sensitivity Analysis

We could always re-solve the linear program with new parameter values. Linear programming solutions are usually quite fast, so this is often a very reasonable option. But, here are some simpler tricks! These are also discussed in Chapter 6 of Hillier and Lieberman.

Lagrange Multipliers/Shadow Prices/Dual Prices

The shadow price (λ) associated with each constraint represents the change in the value of the objective function if the RHS constant of the constraint is changed by one unit.

Note: *Beware the sign of these Lagrange multipliers λ .* A slight change in the formulation of the equations or in the solution method will change the sign of the Lagrange multiplier, although the absolute value should remain the same. It is common for two computer solution packages to give different signs for the shadow prices. In practice, I recommend interpreting the sign intuitively and then applying the absolute value for the magnitude. It is easy to intuit the sign; changing the constraint value to "loosen" the constraint should improve the value of the objective function.

These appear as the coefficients in Row 0, under the slack variables, in the final simplex tableau.

Slack Variables

The value of the slack variable represents the amount or the "resource" remaining un-used in the constraint; i.e., the constraint can be tightened by up to the value of the slack variable before the optimal solution is changed. These values are given directly by the simplex solution.

Reduced Cost

The reduced cost is the amount by which the coefficient in the objective function would have to change to make a non-basic decision variable become basic (non-zero). These values are given by the coefficients in Row 0, under the decision variables, in the final simplex tableau.

Ranges in Which the Basis is Unchanged

Remember, the basis is the set of variables with non-zero values.

Often, linear program output will include a "range in which the basis is unchanged." This output indicates how much particular coefficients in the objective function, RHS constants, and occasionally coefficients in the LHS of the constraints can change before one or more of the basic variables must become non-basic, and vice versa.

Within the “range of objective function coefficient values for which the Basis is unchanged”, changes in these coefficient values will change the value of the objective function, but leave the optimal solution values for the decision variables completely unchanged.

Within the “range of the RHS constants for which the Basis is unchanged”, changes in these RHS coefficient values for *binding* constraints will change both the solution values and the objective function value. For *non-binding* constraints, the range on the tightening side should be the same as the slack variable values.

Example of LINDO Output

MIN 5 X1 + 7 X2
 SUBJECT TO
 2) X1 + X2 >= 10000
 3) - 0.2 X1 + 0.1 X2 >= 0
 4) 0.1 X1 - 0.2 X2 <= 0
 END

LP OPTIMUM FOUND AT STEP 2

OBJECTIVE FUNCTION VALUE

1) 63333.3320

VARIABLE	VALUE	REDUCED COST
X1	3333.333252	0.000000
X2	6666.666504	0.000000

ROW	SLACK OR SURPLUS	DUAL PRICES
2)	0.000000	-6.333333
3)	0.000000	-6.666667
4)	1000.000000	0.000000

NO. ITERATIONS= 2

RANGES IN WHICH THE BASIS IS UNCHANGED:

VARIABLE	OBJ COEFFICIENT RANGES		
	CURRENT COEF	ALLOWABLE INCREASE	ALLOWABLE DECREASE
X1	5.000000	2.000000	19.000000
X2	7.000000	INFINITY	2.000000

ROW	RIGHTHAND SIDE RANGES		
	CURRENT RHS	ALLOWABLE INCREASE	ALLOWABLE DECREASE
2	10000.000000	INFINITY	9999.999023
3	0.000000	1000.000000	1000.000000
4	0.000000	INFINITY	1000.000000

The new final tableau has the form:

Equation No.	Z	X_i X_n	S_j S_m	RHS
0	1	$\underline{y}^* \mathbf{A}' - \underline{c}'$	$\underline{y}^* \mathbf{I}$	Z^*
1	0	$\mathbf{B}^* \mathbf{A}'$	$\mathbf{B}^* \mathbf{I}$	$\mathbf{B}^* \underline{b}'$
m				

Case 2: If the optimal solution has changed corner-points:

(Basic variables have changed.)

The new simplex tableau found in Case 1 will no longer satisfy the optimality test, if the optimal corner-point solution has changed.

This happens if either another corner-point is optimal or there is now no feasible solution.

Using these insights to find changes in parameter values that keep the set of basic variables the same.

Case 3: If the solution is infeasible:

Some basic variable(s) will have negative values. I.e., a decision or slack variable will have a negative value. This can be quickly found because at least one element of $\mathbf{B}^* \underline{b}'$ will be negative.

Duality

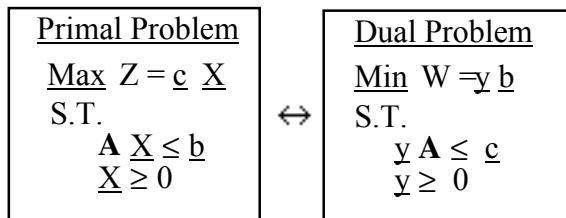
Duality is part of the mathematical theory behind linear programming that makes linear programming a power problem-solving tool.

Some useful results of duality:

- 1) Helps interpretation of LP results.
- 2) Very useful for sensitivity analysis
- 3) Can be used to improve computational efficiency (reducing the number of constraints; simplex solves faster with more decision variables and fewer constraints)
- 4) Useful for speeding multi-objective analysis

What is duality?

For every original (primal) LP problem, there is an associated, dual problem.
Compare the LP statements:



At \underline{X}^* , $Z = \underline{c} \underline{X}$

At \underline{y}^* , $W = \underline{b} \underline{y}$ \therefore At optimal solutions $\underline{c} \underline{X}^* = \underline{b} \underline{y}^* = Z^*$

Compare at Initial Tableaux:

Primal: (Max)

X_i	S_j	RHS
$-\underline{c}$	$\underline{0}$	0
\underline{A}	\underline{I}	\underline{b}

Dual: (Min)

y_i	SD_i	RHS
$-\underline{b}$	$\underline{0}$	0
\underline{A}^T	\underline{I}	\underline{c}

Constraints \Leftrightarrow Decision Variables

Objective function coefficients. \Leftrightarrow RHS

Compare the Final Tableaux:

Primal:

X_i	S_i	
$y^* \underline{A} - \underline{c}$	y^*_i	Z^*
		X^*_i
		S^*_j

Dual:

y_i	SD_i	
$\underline{A} \underline{x}^* - \underline{b}$	X^*_i	Z^*
		y^*_j
		SD^*_j

Note: Solving either the dual or the primal by simplex solves both problems.

The relationship between primal and dual linear programming problems is shown by writing them as (Quirino Paris, n.d.):

<u>Primal</u>		<u>Dual</u>
$\text{Max } Z = c_1x_1 + c_2x_2 + c_3x_3$		$\text{Min } R = b_1y_1 + b_2y_2$
Subject to:		Subject to:
$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \leq b_1$	\longleftrightarrow	$y_1 \geq 0$
$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 \leq b_2$	\longleftrightarrow	$y_2 \geq 0$
$x_1 \geq 0$	\longleftrightarrow	$a_{11}y_1 + a_{21}y_2 \geq c_1$
$x_2 \geq 0$	\longleftrightarrow	$a_{12}y_1 + a_{22}y_2 \geq c_2$
$x_3 \geq 0$	\longleftrightarrow	$a_{13}y_1 + a_{23}y_2 \geq c_3$

Here, y_1 and y_2 are **dual variables** for the **primal constraints** and x_1, x_2, x_3 are **dual variables** for the **dual constraints**. The notion that **Lagrange multipliers** are in essence dual variables can be illustrated by defining the **Lagrangian function** for both linear programming problems. The use of the Lagrangian function in this context is of great help in understanding why and how the primal and the dual LP problems are so intimately related to each other.

We now state "two" Lagrangian functions corresponding to the primal and dual LP problems and conclude that the two functions are identical and, therefore, only one Lagrangian function exists for a pair of primal and dual linear programming problems. We use y_1, y_2 as Lagrange multipliers for the primal problem and x_1, x_2, x_3 , as Lagrange multipliers for the dual.

Lagrangian for Primal		Lagrangian for Dual
$\begin{aligned} \max L = & c_1x_1 + c_2x_2 + c_3x_3 \\ & + y_1[b_1 - a_{11}x_1 - a_{12}x_2 - a_{13}x_3] \\ & + y_2[b_2 - a_{21}x_1 - a_{22}x_2 - a_{23}x_3] \end{aligned}$	$ $	$\begin{aligned} \min L' = & b_1y_1 + b_2y_2 \\ & + x_1[c_1 - a_{11}y_1 - a_{21}y_2] \\ & + x_2[c_2 - a_{12}y_1 - a_{22}y_2] \\ & + x_3[c_3 - a_{13}y_1 - a_{23}y_2] \end{aligned}$

The two Lagrangian functions just stated are actually the same. In this way, the primal and the dual LP problems are tied together by a unique Lagrangian function.

Duality Example:

For the Primal Problem:

$$\text{Max } Z = 3X_1 + 5X_2$$

Subject to:

$$X_1 + X_2 \leq 10$$

$$2X_1 + X_2 \leq 12$$

$$X_1 \leq 4$$

Find the Dual Problem:

$$\text{Min } Z = 10Y_1 + 12Y_2 + 4Y_3$$

Subject to:

$$Y_1 + 2Y_2 + Y_3 \geq 3$$

$$Y_1 + Y_2 \geq 5$$

An economic interpretation: If the primal problem is to maximize profit subject to resource constraints, we can interpret Y_j as the *contribution to profit* per unit of resource j . The objective of the dual problem, then, is to minimize the *total implicit value of the resources consumed*.

$Y_j = \lambda_j =$ the marginal contribution to profit of resource j .

Some Primal-Dual Relationships for LP

1) Weak Duality Property

If both \underline{X}^o and \underline{Y}^o are complementary feasible solutions for the primal and dual problems, respectively, then:

$$\underline{c} \underline{X}^o \leq \underline{Y}^o \underline{b} .$$

2) Strong Duality Property

If both \underline{x}^* and \underline{y}^* are complementary optimal solutions for the primal and dual problems, respectively, then:

$$\underline{c} \underline{X}^* = \underline{Y}^* \underline{b} .$$

Note: Computationally, we would rather have variables than constraints.

B* summarizes all the operations performed on the simplex tableau to get the final solution.

The Final Tableau:

Basic Var.	Eqn. No.	Coefs. of Variables				RHS
		Z	X_i	X_n	S_j S_m	
Z	0	1	$[\underline{y}^* \underline{A}' - \underline{c}']$		$[\underline{y}^*]$	$[Z^*]$
	1	0	$[\underline{A}^*]$		$[\underline{B}^*]$	$[\underline{b}^*]$
	.					
	.					
	m					

Initial Tableau: The initial tableau consists of the stacked vector and matrix, $\begin{bmatrix} \underline{t} \\ \underline{T} \end{bmatrix}$
 \underline{t} = vector of Row 0 in initial tableau $[\underline{c} \mid \underline{0} \mid 0]$

\underline{T} = matrix of Rows 1 to m in the initial Tableau $[\underline{A} \mid \underline{I} \mid \underline{b}]$

Final Tableau: The final tableau consists of the stacked vector and matrix, $\begin{bmatrix} \underline{t}^* \\ \underline{T}^* \end{bmatrix}$

$\underline{t}^* = [\underline{Z}^* = \underline{y}^* \underline{A}]$, a consequence of operations done to Row 0

$$\underline{T}^* = [\underline{A}^* \mid \underline{B}^* \mid \underline{b}^*]$$

$$\underline{A}^* = \underline{B}^* \underline{A} \quad \underline{b}^* = \underline{B}^* \underline{b}$$

"Fundamental Insight" (?)

$$\underline{t}^* = \underline{t} + \underline{y}^* \underline{T} = [\underline{y}^* \underline{A} = \underline{c} \mid \underline{y}^* \mid \underline{y}^* \underline{b}]$$

$$\underline{T}^* = \underline{B}^* \underline{T} = [\underline{B}^* \underline{A} \mid \underline{B}^* \mid \underline{B}^* \underline{b}]$$

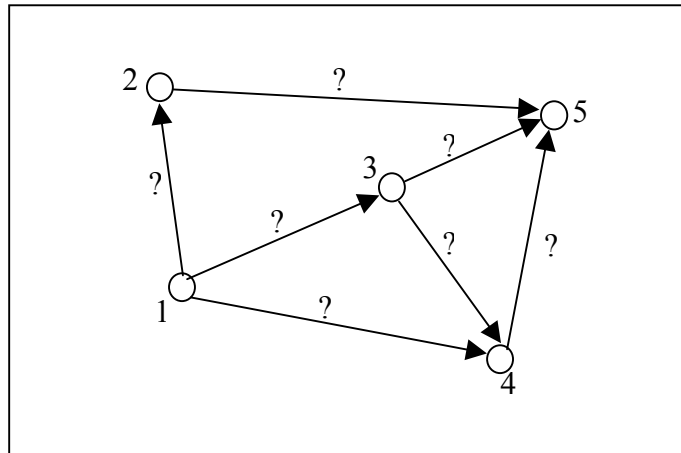
Transportation Problems

A Simple Transportation Problem

Common problem of shipping good between multiple sources and destinations. Minimize cost.

- Let:
- n = number of locations
 - X_{ij} = units of good (flow) shipped from origin i and destination j
 - c_{ij} = Cost of transporting a unit of good from i to j
 - d_j = demand for good at each location j
 - s_i = supply of good at each source location i

Here's the Problem. How much flow should go where?



What's the LP?

$$\begin{aligned} \text{Min } Z &= \sum_{i=1}^n \sum_{j=1}^m c_{ij} X_{ij} \\ \text{S.T. } \sum_{i=1}^n X_{ij} &\geq d_j, j = 1, \dots, n \text{ (satisfy demands at ea. location)} \\ \sum_{j=1}^n X_{ij} &\leq s_i, i = 1, \dots, n \text{ (limited sources at each location)} \\ X_{ij} &\geq 0, \forall i \text{ and } j \end{aligned}$$

There are n^2 decision variables and $2n$ constraints for this problem.
 Project homework is a transportation problem, plus a little bit more.

Multi-Period Transportation Problem with Warehouse Storage

Add time subscripts to each variable & coefficient to represent each of T time-periods in the plan.

- Let:
- T = number of scheduling periods.
 - $X_{ijt}, d_{jt}, s_{it}, c_{ijt}$
 - c_{wit} = unit cost of warehousing at location i and time t
 - K_{kt} = Storage capacity of warehouse at location k and time t .

This problem is a little more complex. Since warehouses store goods in time, a time dimension (subscript) is essential to this problem.

$$\text{Min } Z = Z = \sum_{t=1}^T \left[\sum_{i=1}^n \sum_{j=1}^n c_{ijt} X_{ijt} + \sum_{i=1}^n c_{wit} W_{it} \right]$$

Subject To

$$\sum_{i=1}^n X_{ijt} + W_{jt} + S_{jt} = \sum_{k=1}^n X_{jkt} + D_{jt} + W_{j,t+1} \quad \forall j,t \text{ (conservation of mass at all locations and times)}$$

$$D_{jt} \geq d_{jt}, \quad \forall j,t \text{ (satisfy demands at all locations and times)}$$

$$S_{it} \leq s_{it}, \quad \forall i,t \text{ (limited sources at all locations and times)}$$

$$W_{kt} \leq b_{kt}, \quad \forall k,t \text{ (limited storage capacity)}$$

$$X_{ijt} \leq r_{ijt}, \quad \forall i,j,t \text{ (limited route capacity on each route at each time)}$$

$$X_{ijt}, X_{kjt}, X_{ikt} \geq 0, \quad \forall i,j,k,t \text{ (non-negativity)}$$

Number of decision variables = Tn^2
 Number of constraints = $(4n + n^2)T$
 Number of parameters to estimate (c, c_w, b, r, d, and s) = $2n^2 + 4n$

Transportation with Warehouses and Travel Times/Delivery Lags

Let: X_{ijt} = departures from i to j at time t
 A_{ijt} = arrivals from i to j at time t
 L_{ijt} lag or travel time from i to j departing at time t
 Note: $X_{ijt} = A_{ij,t+L_{ijt}}$ since departures at time t arrive at time t + L_{ijt} .

The LP?

$$\text{Min } Z = Z = \sum_{t=1}^T \left[\sum_{i=1}^n \sum_{j=1}^n c_{ijt} X_{ijt} + \sum_{i=1}^n c_{wit} W_{it} \right]$$

Subject To

$$\sum_{i=1}^n A_{ijt} + W_{jt} + S_{jt} = \sum_{k=1}^n X_{jkt} + D_{jt} + W_{j,t+1} \quad \forall j,t \text{ (conservation of mass at all locations and times)}$$

$$X_{ijt} = A_{ij,t+L_{ijt}} \quad \forall i,t \text{ (arrivals lag departures by } L_{ijt})$$

$$D_{jt} \geq d_{jt}, \quad \forall j,t \text{ (satisfy demands at all locations and times)}$$

$$S_{it} \leq s_{it}, \quad \forall i,t \text{ (limited sources at all locations and times)}$$

$$W_{kt} \leq b_{kt}, \quad \forall k,t \text{ (limited storage capacity)}$$

$$X_{ijt} \leq r_{ijt}, \quad \forall i,j,t \text{ (limited route capacity on each route at each time)}$$

$$X_{ijt}, X_{kjt}, X_{ikt} \geq 0, \quad \forall i,j,k,t \text{ (non-negativity)}$$

This “transportation” problem is now rather flexible and applies to lots of problems. Variations of this also can apply to other logistics problems, such as water and wastewater movements – Just consider reservoirs to be warehouses and you can model California’s water system this way.

Scheduling Problems

Simple Scheduling Problems

CPM and related problems.

There are n activities to schedule to complete a project.

- Let:
- S_i = the starting time of activity
 - d_i = duration of the activity
 - p_{ij} = 1 if activity j is a prerequisite of activity i ; equals zero otherwise.

Objective: Min. time to completion = T :

The LP:

$$\begin{array}{ll} \text{Min} & T \\ \text{S.T.} & T \geq S_i + d_i, \quad i = 1, \dots, n \\ & S_i \geq p_{ij}S_j + p_{ij}d_j, \quad i = 1, \dots, n \text{ and } j = 1, \dots, n \end{array}$$

Scheduling where Activity Duration can be shortened at a cost:

Fundamentally, this is a *multi-objective* optimization problem (Cohon 1978). The two objectives are 1) minimize cost and 2) minimize time to completion T . A weighting factor α can be assigned to combine and weight these two objectives in the overall optimization objective function.

↓ is max duration of activity

Let $d_i = c_i - a_i X_i \leftarrow X_i = \$$ spent to speed completion of activity i

$d_i \geq b_i \leftarrow$ minimum duration of activity

$$\begin{array}{ll} \text{Min} & Z = \sum_{i=1}^n X_i + \alpha T \\ \text{S.T.} & T \geq S_i + d_i, \quad \forall i \\ & S_i \geq p_{ij}S_j + p_{ij}d_j, \quad \forall i, j, i \neq j \\ & d_i \geq b_i, \quad \forall i \\ & d_i = c_i - a_i X_i, \quad \forall i \end{array}$$

Minimize a combination of Cost and Time-duration

Weighting Method:

Here, we'll weight each of the two objectives. Making several LP runs with different weights will produce an efficient trade-off curve for the two objectives.

Change objective function to:

$$\text{Min } \alpha \left(\sum_{i=1}^n X_i \right) + (1 - \alpha) T \quad .$$

Constraint Method:

Another multi-objective optimization approach is the "constraint method". This would involve solving the cost-minimizing LP several times, with each LP run having a different constrained value for the completion time T , $T \leq \alpha$. The constant α will vary from T_{\min} (resulting from the completion-time minimizing formulation) to T_{\max} (the cost-minimizing solution). This "constraint method" is actually a bit more rigorous.

Assignment Problems

Allocating resources on a one-to-one basis.

Example allocating personnel to tasks on a project so as to 1) maximize overall performance or 2) make employees happier.

Example: Allocate m graduate students to m available desks. Students have given preferences for particular desks (1 = most preferred, ..., m = least preferred).

c_{ij} = student i 's preference for desk j .

X_{ij} = 1 if student i sits in desk j .

X_{ij} = 0 if not.

The Linear Program version of this problem:

$$\begin{array}{ll} \text{Min} & \sum_{i=1}^m \sum_{j=1}^m c_{ij} X_{ij} \\ \text{S.T.} & \sum_{i=1}^m X_{ij} = 1, \text{ for } j = 1, \dots, m \\ & \sum_{j=1}^m X_{ij} = 1, \text{ for } i = 1, \dots, m \end{array}$$

Number of decision variables = m^2

Number of constraints = $2m$.

Other LP problems

[Should include some more of these examples.]

Regional Planning

Reservoir operations

Water Quality

Regional environmental planning

Air Quality Planning

Product Mix

Goal Programming

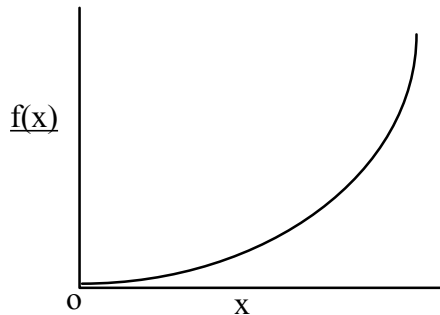
[Read text](#)

LP Solution of Non-Linear Objectives with Linear Constraints

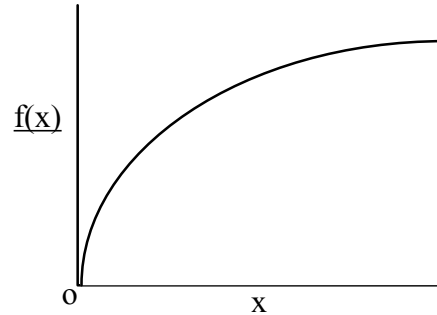
Sounds contradictory. But can be done for problems with objective functions of forms:

Min $f(x)$: (convex)

Max $f(x)$: (concave)



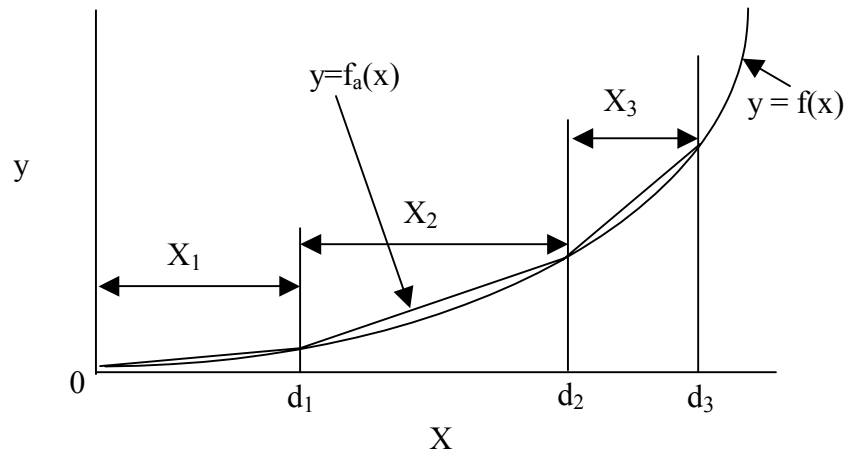
or



Piece-wise Linearization

In each case, a linear approximation can be made of the function for specific ranges of X :

Min $f(x)$:



$$f_a(X) \approx f(X)$$

The resulting *linearized* non-linear program is:

$$\begin{aligned} \text{Min } f_a(X) = & f(0) + \frac{f(d_1) - f(0)}{d_1} X_1 \\ & + \frac{f(d_2) - f(d_1)}{d_2 - d_1} X_2 \\ & + \frac{f(d_3) - f(d_2)}{d_3 - d_2} X_3 \end{aligned}$$

Subject to:

$$X = X_1 + X_2 + X_3$$

$$X_1 \leq d_1$$

$$X_2 \leq d_2 - d_1$$

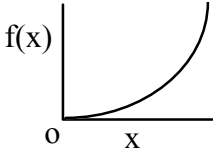
$$X_3 \leq d_3 - d_2$$

followed by any other constraints

Note: The coefficients of X_1 , X_2 , and X_3 are just slopes of parts of the objective function!

Why it works:

LP will always make $X_1 = d_1$ before X_2 becomes > 0 .
 Similarly for X_2 and X_3 .

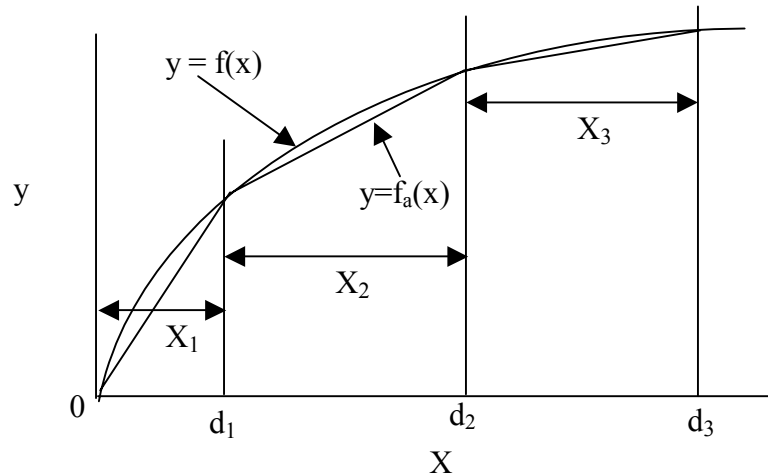
To maximize this same function,  it won't work.

The LP will want to max X_3 and keep X_1 and $X_2 = 0$.

Maximizing Concave Objective Functions

A similar linearization approach will work for maximizing a concave function subject to linear constraints.

Max $f(X)$:



Later, we will see how any type of single-decision-variable can be linearized, with the addition of integer-linear programming.

[Need a numerical example.]

Example of Piecewise Linearization

- 1) Minimize a convex function,
 $\text{Min } Z = 5X^2$
 S.T. constraints on X
- 2) Maximize a concave function,
 $\text{Max } Z = 3X^{0.7}$
 S.T. constraints on X

Integer Programming and Mixed Integer-Linear Programming

For many problems, the decision variables can only take on integer values.

For example, for a location decision, typically a facility can be located at only one site, a factory can sell only an integer number of airplanes, etc.

Decision variables with integer values only are called "integer variables".

A common case is where a decision variable can have values of either 0 or 1. These are called "binary variables". "Yes" or "no" values for variable.

Example: Capital Improvement Plan

A city sewer utility must decide which capital improvement projects to select to minimize long-term operation and maintenance costs.

Decision No.	Decision	Decision Variable	Net Present Value	Capital Requirements
1	Sewer Rehab. #1	X_1	\$70 million	\$20 m
2	WWTP Rehab. #1	X_2	\$50 million	\$15 m
3	Sewer Rebuild #1	X_3	\$40 million	\$12 m
4	Sewer Rehab#2	X_4	\$30 million	<u>\$10 m</u>
			Capital Avail.	\$25 million

Cannot build everywhere because of capital availability constraint.
 X_i = yes or no, 0 = no, 1 = yes.

Max $Z = 70X_1 + 50X_2 + 40X_3 + 30X_4$ ← max. total new present value
 S.T. 1) $20X_1 + 15X_2 + 12X_3 + 10X_4 \leq 25$ ← capital constraint
 2) X_1, X_2, X_3, X_4 are binary (0,1)

Now to solve?

1) By enumeration: Try all combinations of $X_i = 0, 1$. Choose combination with the best feasible value of the objective function.

For example: $2^4 = 16$ combinations, for n binary decision variables where are 2^n combinations.
 Gets to be lots of combinations fast.

2) Branch and Bound Methods: The concept:

An efficient enumeration procedure.

a) For the example, we can see that any solution where $X_1 = 1$ and $X_2 = 1$ is infeasible, violating the budget constraint.

For these values of X_1 and X_2 , the solution is infeasible regardless of the values of X_3 and X_4 .

So, by finding an infeasible subset of decision values, we eliminate many combinations of the larger set of values. Here, we eliminated: (1,1,0,0), (1,1,0,1), (1,1,1,0), and (1,1,1,1).

b) Similarly, we can eliminate combinations of decision values which lead to inferior solutions.

For the example:

Let the decision be (0,1,0,1). This is feasible (just barely). Since the objective is one of maximization and the decrease in any decision variable only decreases the objective function, we know that (0,1,0,1) is better than (0,0,0,1), (0,0,0,0), and (0,1,0,0).

The book and appendix detail methods for putting these ideas to efficient practice.

Typically, a branch and bound approach will significantly limit the number of combinations to be evaluated. (This can still be a lot.)

Mixed Integer-Linear Programming

Often we must solve LP problems that have one or more binary or integer decision variables. The Hillier and Lieberman goes over several Branch-and-Bound approaches (pp. 515-541). Some major concepts for understanding such algorithms are:

- LP relaxation
- Branching
- Bounding
- Fathoming

Appendix VIII provides a simple algorithm for solving integer-linear programs.

Summary of the BIP Branch and Bound Algorithm

[This section needs some work.]

Initialization. Set $Z^* = -\infty$ (for a maximization problem). Apply the bounding step, fathoming step, and optimality test described below to the entire problem. If not fathomed, classify this problem as the one remaining “subproblem.”

Steps for each iteration:

1. *Branching.* Among the remaining (*unfathomed*) subproblems, select the one that was created most recently. (Break ties according to which one has the larger bound). Branch from the node for this problem to create two new subproblems by fixing the next variable (the *branching variable*) to 0 or 1.
2. *Bounding.* For each new subproblem, obtain its *bound* by applying the simplex method to its *LP relaxation*.

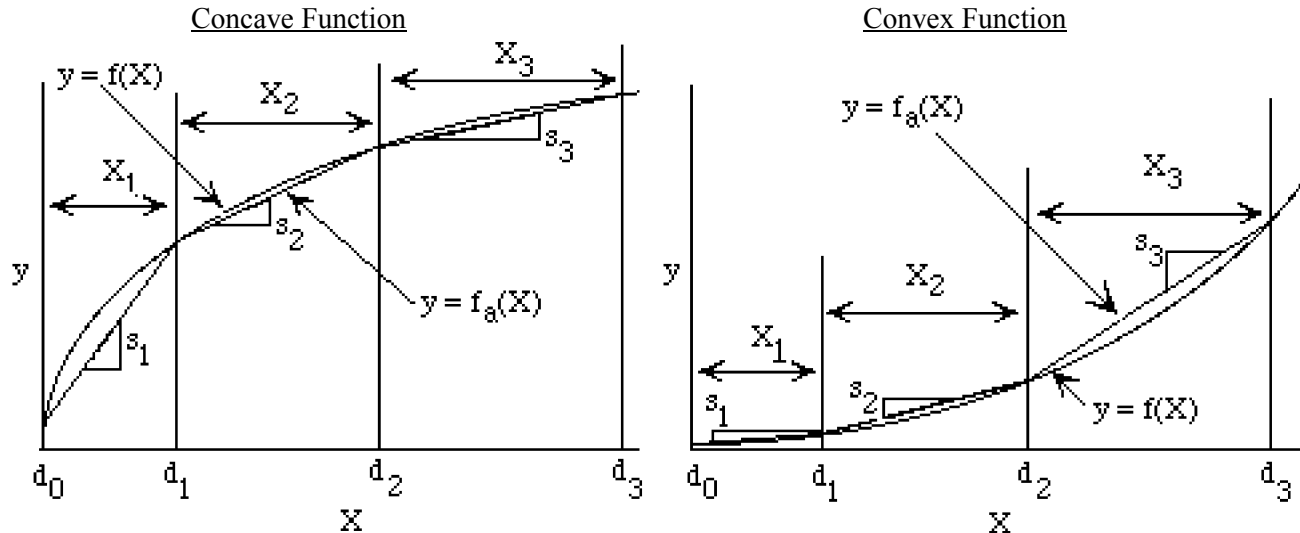
[This section needs some work.]

3. *Fathoming.* For each new subproblem, apply the three fathoming tests described below, and discard those subproblems that are fathomed by any of the tests. A subproblem is fathomed (dismissed from further consideration) if:
 - Its bound is less than or equal to Z^* (i.e., we already know of a better solution).
 - Its LP relaxation has no feasible solutions.
 - The optimal solution to its LP relaxation satisfies the integer constraints. (If this solution is better than the incumbent, it becomes the new incumbent.)

Optimality test: If there are *no remaining subproblems*, STOP. The incumbent is optimal. Otherwise, perform another iteration.

More Non-Linear Objective Functions with Linear Constraints Using Integer-Linear Programming.

Loucks, et al. (1981) have a nice presentation of linearizing any single-variable objective function, although often integer-linear programming is required. Consider the two functions:



Method for Linear Programming (Max. concave or Min. convex objective functions)

The first method of linearization is appropriate for linear programming and was described above. It is restated below.

Maximize $f(X) = s_1X_1 + s_2X_2 + s_3X_3 = \sum_{i=1}^n s_iX_i$ (for the concave function only)

Minimize $f(X) = s_1X_1 + s_2X_2 + s_3X_3 = \sum_{i=1}^n s_iX_i$ (for the convex function only)

Subject to: $X = d_0 + X_1 + X_2 + X_3$
 $X_j \leq d_j - d_{j-1}$, for each linear segment j
 and any original linear constraints in the problem

Method for Mixed Integer-Linear Programming (either convex or concave functions)

Max or Min $f(X) \approx f(d_0)Z_0 + f(d_1)Z_1 + f(d_2)Z_2 + f(d_3)Z_3 + s_1X_1 + s_2X_2 + s_3X_3$

$$= \sum_{i=0}^n [f(d_{i-1})Z_{i-1} + s_iX_i]$$

Subject to: $X = d_0Z_0 + d_1Z_1 + d_2Z_2 + d_3Z_3 + X_1 + X_2 + X_3$
 $Z_0 + Z_1 + Z_2 + Z_3 = 1$
 $X_j \leq (d_j - d_{j-1})Z_{j-1}$ for each linear segment j
 Z_j are integer variables for all j
 and any original linear constraints in the problem

Computational Aspects

It is often possible to formulate problems that are too large for computers to solve. How large a problem can we formulate before we have to start worrying?

Computational Aspects of Integer Programs

Wagner (1975) (p. 490)

-) Fewer than 100 binary variables can be solved "without great danger"
-) "Real problems containing actual data often solve more rapidly than you might predict on theoretical grounds."

Hillier and Lieberman (1986) (p. 400)

-) "Even the fastest computers are incapable of performing exhaustive enumeration for binary integer problems with a few dozen variables..."

LP Computation Requirements

Schrage (p. 186)

$$\begin{aligned} \text{LP effort} &\propto \text{No. of decision variables} \\ &\propto (\text{No. of constraints})^2 \end{aligned}$$

Hillier and Lieberman (1986) (p. 92)

- LP effort: a) is less than proportional to the number of decision variables
- b) is roughly proportional to (No. of constraints)³
- c) decreases with increasing "sparseness" of the constraint matrix (**A**)

Problems with 5,000 constraints and 10,000 decision variables can be solved in < 1 hr.

Recent LP Solver Experiences (1999): Problems with > 20,000 decision variables solved in 3 minutes.

Specialized LP problems, such as Network Flow Programs:

Solution time for some common specialized problems tends to increase about proportionately with the number of constraints squared or less.