MCSD1133 Operations Research & Optimization

UTM
UNIVERSITI TEKNOLOGI MALAYSIA

CHAPTER 7: ILP – TRANSPORTATION MODEL

UTM JOHOR BAHRU

nzah@utm.my-20232024(S2)

Innovating Solutions

The Transportation Model

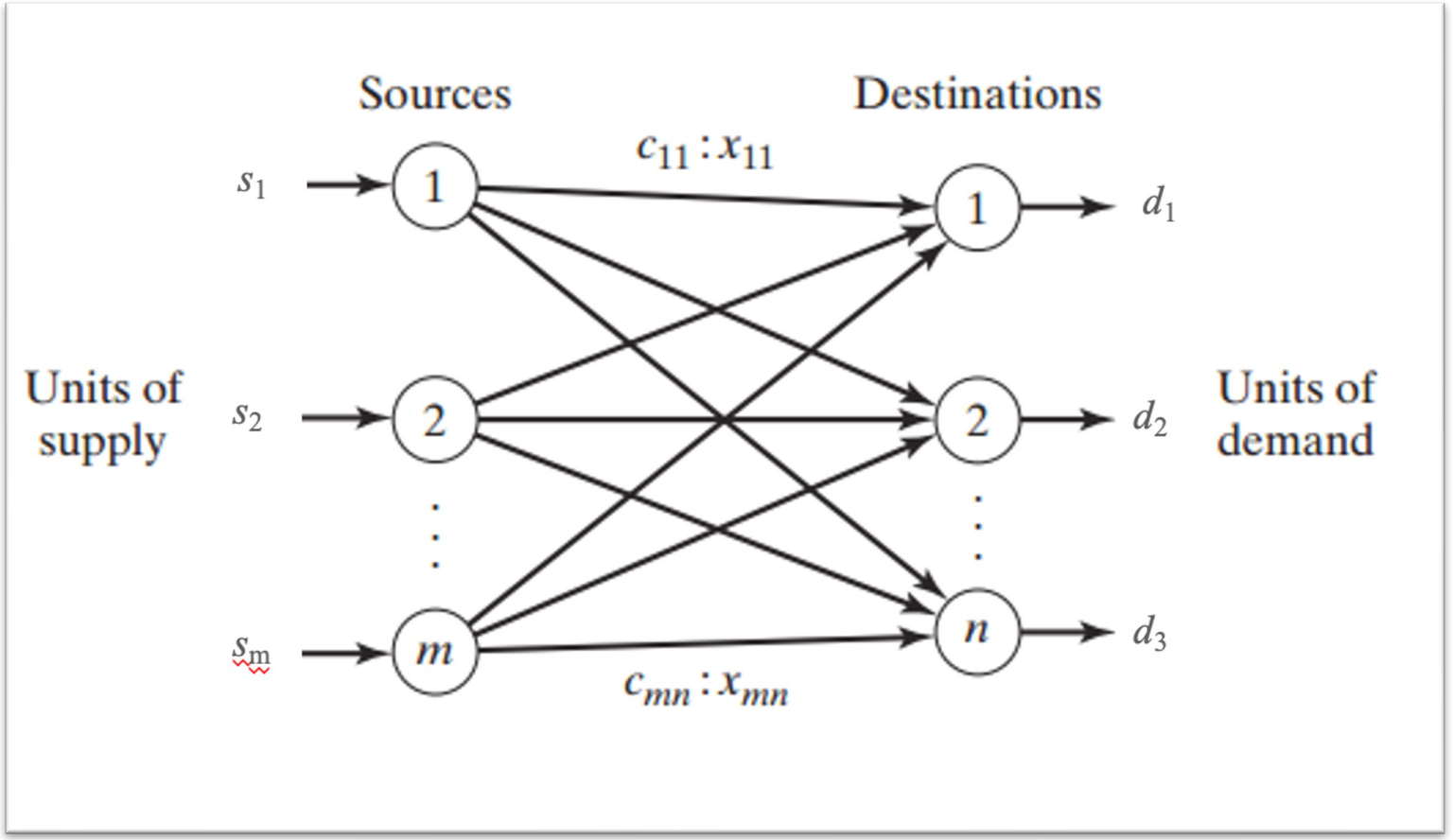
- A product is transported from a few sources to several destinations at the ***minimum possible cost***.
- Each ***source can supply a fixed number*** of units of the product, and ***each destination has a fixed demand*** for the product.
- The linear programming model has ***constraints for supply*** at each source ***and demand*** at each destination.
- All constraints are equalities in a balanced transportation model where ***supply equals demand*.**
- Constraints contain inequalities in unbalanced models where supply does not equal demand.

The Transportation Model

- There are m sources and n destinations, each represented by a **node**. The **arcs** (i, j) represent the routes linking the sources and the destinations.
- To develop a model of a transportation problem, it is necessary to have the following information:
    - Supply quantity (capacity) at source i, s_i, ,.
    - Demand quantity of each destination j, d_j.
    - Arc (i, j) joining source i to destination j carries two pieces of information: the transportation cost per unit, c_{ij}, and the amount shipped, x_{ij}.
- The objective of the model is to minimize the total transportation cost while satisfying all the supply and demand restrictions.

The Transportation Model

Schematic of a Transportation Problem



The Transportation Model

- Transportation problem is considered such an important special type of linear programming problem.
- For many applications, the supply and demand quantities in the model (the s_i and d_j) have integer values, and implementation will require that the distribution quantities (the x_{ij}) also have integer values.
- The solution procedure deals only with basic feasible solutions, so it automatically will obtain an ***integer*** optimal solution for this case. Therefore, it is unnecessary to add a constraint to the model that the x_{ij} must have integer values.

The Transportation Algorithm

The basic steps of the transportation algorithm are *exactly* those of the simplex method. However, instead of using the regular simplex tableau, we take advantage of the special structure of the transportation model to carry out the algorithmic computations more conveniently.

Step 1. Determine a *starting* basic feasible solution using one of the three methods.

$$m + n - 1 = \text{basic variables}$$

After determining the starting solution, use the following algorithm to determine the optimum solution:

Step 2: Use the optimality condition of the simplex method to determine the *entering variable* from among all the non-basic variables. If the optimality condition is satisfied, stop. Otherwise, go to Step 3.

Step 3. Use the feasibility condition of the simplex method to determine the *leaving variable* from among all the current basic variables and find the new basic solution. Return to Step 2.

Example 1

SunRay Transport Company ships truckloads of grain from three silos to four mills. The supply (in truckloads) and the demand (also in truckloads) together with the unit transportation costs per truckload on the different routes are summarized in Table 1.

Table 1

	Mill 1	Mill 2	Mill 3	Mill 4	Supply
Silo 1	10 x_{11}	2 x_{12}	20 x_{13}	11 x_{14}	**15**
Silo 2	12 x_{21}	7 x_{22}	9 x_{23}	20 x_{24}	**25**
Silo 3	4 x_{31}	14 x_{32}	16 x_{33}	18 x_{34}	**10**
Demand	**5**	**15**	**15**	**15**	

The unit transportation costs, c_{ij} (shown in the northeast corner of each box), are in hundreds of dollars. The model seeks the minimum cost shipping schedule between the silos and the mills.

STEP 1: Determination of the Starting Basic Solution

- The special structure of the transportation problem allows securing a non-artificial starting basic solution using one of three methods:
    i. Northwest-corner method
    ii. Least-cost method
    iii. Vogel approximation method
- The first method is “mechanical” in nature in that its main purpose is to provide a starting (basic feasible) solution regardless of the cost.
- The remaining two are heuristics that seek a better-quality (smaller objective value) starting solution. In general, the Vogel heuristic is best, and the northwest-corner method is the worst.

Northwest-Corner Method

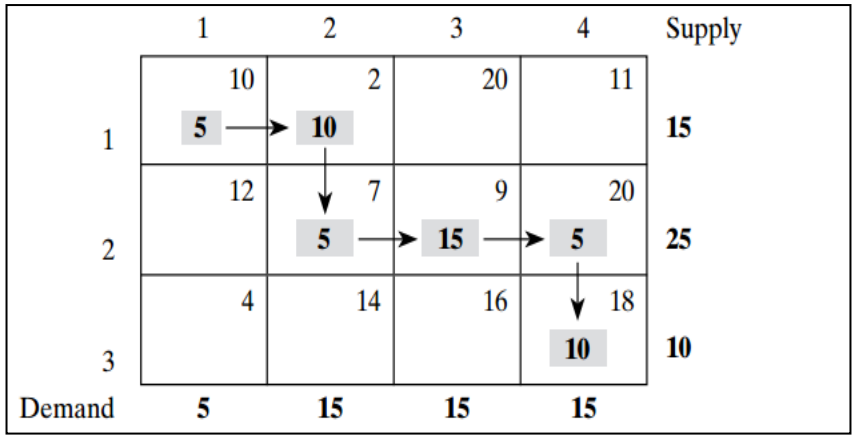
The method starts at the northwest-corner cell (route) of the tableau (variable $x11$)

Step 1. Allocate as much as possible to the selected cell, and adjust the associated amounts of supply and demand by subtracting the allocated amount.

Step 2. Cross out the row or column with zero supply or demand to indicate that no further assignments can be made in that row or column. If both a row and a column net to zero simultaneously, *cross out one only*, and leave a zero supply (demand) in the uncrossed-out row (column).

Step 3. If *exactly one* row or column is left uncrossed out, stop. Otherwise, move to the cell to the right if a column has just been crossed out or below if a row has been crossed out. Go to step 1.

Northwest-Corner Method



	1	2	3	4	Supply
1	10 **5** →	2 **10**	20	11	15
2	12	7 **5** →	9 **15** →	20 **5**	25
3	4	14	16	18 **10**	10
Demand	5	15	15	15	

The starting basic solution is
$x11 = 5, x12 = 10, x22 = 5, x23 = 15, x24 = 5, x34 = 10.$

The associated cost of the schedule is
$z = 5 * 10 + 10 * 2 + 5 * 7 + 15 * 9 + 5 * 20 + 10 * 18$
$= \$520$

- The first allocation is x11 = 5, which exactly uses up the demand in column 1, and cross out column 1.
- This first iteration leaves a supply of 10 (15-5) remaining in row 1, so next move to the cell to the right x1,1+1 = x12 and allocate x12=10. Then, cross out row 1.
- Next, move to the cell to the below x1+1, 2 = x22 and allocate x22 = (15-10 = 5). Cross out column 2.
- Move to the cell to the right x2, 2+1 = x23 and allocate x23 = 15. Cross out column 3.
- Move to the cell to the right x2, 3+1 = x24 and allocate x24 = 5
- Allocate x34 = 10 and stop.

Least-Cost Method

Finds a better starting solution by targeting the cheapest routes.

Step 1: Assigns as much as possible to the cell with the smallest unit cost.

Step 2: Next, the satisfied row or column is crossed out and the amounts of supply and demand are adjusted accordingly. If both a row and a column are satisfied simultaneously, only one is crossed out, the same as in the northwest-corner method.

Step 3: Select the uncrossed-out cell with the smallest unit cost and repeat the process until exactly one row or column is left uncrossed out.

Least-Cost Method

1. Cell (1, 2) has the least unit cost in the tableau 1 ($2). The most that can be shipped through (1, 2) is x_{12} = 15 truckloads, which happens to satisfy both row 1 and column 2 simultaneously. We arbitrarily cross out column 2 and adjust the supply in row 1 to 0.
2. Cell (3, 1) has the smallest uncrossed-out unit cost ($4). Assign x_{31} = 5, and cross out column 1 because it is satisfied, and adjust the demand of row 3 to 10 - 5 = 5 truckloads.
3. Continuing in the same manner, we successively assign 15 truckloads to cell (2, 3), 0 truckloads to cell (1, 4), 5 truckloads to cell (3, 4), and 10 truckload to cell (2, 4)

	Mill 1	Mill 2	Mill 3	Mill 4	Supply
1	10 x_{11}	2 x_{12}	20 x_{13}	11 x_{14}	**15**
Silo 2	12 x_{21}	7 x_{22}	9 x_{23}	20 x_{24}	**25**
3	4 x_{31}	14 x_{32}	16 x_{33}	18 x_{34}	**10**
Demand	**5**	**15**	**15**	**15**	

Least-Cost Method

- The resulting starting solution is summarized in Table 2. The arrows show the order in which the allocations are made.
- The starting solution (consisting of six basic variables (3 + 4 - 1) is $x_{12} = 15$, $x_{14} = 0$, $x_{23} = 15$, $x_{24} = 10$, $x_{31} = 5$, $x_{34} = 5$.
- The associated objective value is
  $z = (15 * \$2) + (0 * \$11) + (15 * \$9) + (10 * \$20) + (5 * \$4) + (5 * \$18) = \$475$

Table 2

	1	2	3	4	Supply
1	10	(start) 2 **15**	20	11 **0**	15
2	12	7	9 **15**	(end) 20 **10**	25
3	4 **5**	14	16	18 **5**	10
Demand	5	15	15	15	

Vogel Approximation Method

Step 1:

- For each row (column), determine a *penalty measure* by subtracting the *smallest* unit cost in the row (column) from the *next smallest* unit cost in the same row (column).
- This penalty is actually a measure of *lost opportunity* one forgoes if the smallest unit cost cell is not chosen.

Table 3

	1	2	3	4		Row penalty
1	10	2	20	11	**15**	$10 - 2 = 8$
2	12	7	9	20	**25**	$9 - 7 = 2$
3	4	14	16	18	**10**	$14 - 4 = 10$
	5	**15**	**15**	**15**		
Column penalty	$10 - 4 = 6$	$7 - 2 = 5$	$16 - 9 = 7$	$18 - 11 = 7$		

Vogel Approximation Method

Step 2:

- Identify the row or column with the largest penalty, breaking ties arbitrarily. Allocate as much as possible to the variable with the least unit cost in the selected row or column.
- Adjust the supply and demand and cross out the satisfied row *or* column. If a row and a column are satisfied simultaneously, only one of the two is crossed out, and the remaining row (column) is assigned zero supply (demand).
- Because row 3 has the largest penalty (10) and cell (3, 1) has the smallest unit cost in that row, the amount 5 is assigned to x_{31}. Column 1 is now satisfied and must be crossed out and leaves 5 units in row 3. Next, new penalties are recomputed as in Table 4.

Table 4

	1	2	3	4		Row pen
1	10	2	20	11	**15**	9
2	12	7	9	20	**25**	2
3	4 **5**	14	16	18	**10**	2
	5	**15**	**15**	**15**		
Column penalty	—	5	7	7		

Vogel Approximation Method

Step 2 (cont'd):

- Next, row 1 has the highest penalty (i.e., 9). Hence, we assign the maximum amount possible to cell (1, 2), which yields $x_{12} = 15$ and simultaneously satisfies both row 1 and column 2.
- Cross out column 2 and adjust the supply in row 1 to zero.
- Continuing in the same manner, row 2 will produce the highest penalty (11), and assign $x_{23} = 15$

	1	2	3	4		Row penalty
1	10	**15** **2**	20	11	**0**	**9**
2	12	**7**	9	20	**25**	**11**
3	4 **5**	**14**	16	18	**5**	2
	5	**15**	**15**	**15**		
Column penalty	—	—	7	7		

Vogel Approximation Method

Step 2 (cont'd):

- Crosses out column 3 and leaves 10 units in row 2.

	1	2	3	4		Row penalty
1	10	2 15	20	11	0	9
2	12	7	9 15	20	10	11
3	4 5	14	16	18	5	2
	5	15	15	15		
Column penalty	—	—	–	7		

Vogel Approximation Method

Step 3:

a) If exactly one row or column with zero supply or demand remains uncrossed out, stop.

b) If one row (column) with *positive* supply (demand) remains uncrossed out, determine the basic variables in the row (column) by the least-cost method. Stop.

c) If all the uncrossed-out rows and columns have (remaining) zero supply and demand, determine the *zero* basic variables by the least-cost method. Stop.

d) Otherwise, go to step 1.

Vogel Approximation Method

Only column 4 is left, and it has a positive supply of 15 units. Applying the least-cost method to that column:

- Assign $x_{14} = 0$, $x_{34} = 5$, and $x_{24} = 10$
- The associated objective value for this solution is

$z = (15 * \$2) + (0 * \$11) + (15 * \$9) + (10 * \$20) + (5 * \$4) + (5 * \$18) = \$475.$

	1	2	3	4		Row penalty
1	10	15 **2**	**20**	11	**0**	**9**
2	12	**7**	**15** **9**	20 **10**	**0**	**11**
3	4 **5**	**14**	**16**	18 **5**	**0**	2
	5	**15**	**15**	**15**		
Column penalty	—	—	–	7		

STEP 2 (Iteration 1): Determine the Entering Basic Variable

In the method of multipliers, the multipliers u_i and v_j is associate with row-*i* and column-*j* of the transportation tableau:

$$u_i + v_j = c_{ij}, \text{ for each } basic$$

then evaluate the non-basic variables by computing:

$$u_i + v_j - c_{ij}, \text{ for each } nonbasic\ x_{ij} \text{ (southeast corner of each cell)}$$

set $u_1 = 0$

	$v_1 = 10$	$v_2 = 2$	$v_3 = 4$	$v_4 = 15$	Supply
$u_1 \equiv 0$	10 **5**	2 **10**	20 −16	11 4	**15**
$u_2 = 5$	12 3	7 **5**	9 **15**	20 **5**	**25**
$u_3 = 3$	4 9	14 −9	16 −9	18 **10**	**10**
Demand	**5**	**15**	**15**	**15**	

$u1 + v1 = 10$; u1=0, v1=10
$u1 + v2 = 2$; u1=0, v2=2
$u2 + v2 = 7$; u2=5, v2=2
$u2 + v3 = 9$; u2=5, v3=4
$u2 + v4 = 20$; u2=5, v4=15
$u3 + v4 = 18$; u3=3, v4=15

Nonbasic variable	$u_i + v_j - c_{ij}$
x_{13}	$u_1 + v_3 - c_{13} = 0 + 4 - 20 = -16$
x_{14}	$u_1 + v_4 - c_{14} = 0 + 15 - 11 = 4$
x_{21}	$u_2 + v_1 - c_{21} = 5 + 10 - 12 = 3$
x_{31}	$u_3 + v_1 - c_{31} = 3 + 10 - 4 = \mathbf{9}$
x_{32}	$u_3 + v_2 - c_{32} = 3 + 2 - 14 = -9$
x_{33}	$u_3 + v_3 - c_{33} = 3 + 4 - 16 = -9$

STEP 2 (Iteration 1): Determine the Entering Basic Variable

Find the Entering Basic Variable. The preceding information, together with the fact that $u_i + v_j - c_{ij} = 0$ for basic x_{ij}, is equivalent to computing the z-row of the simplex tableau.

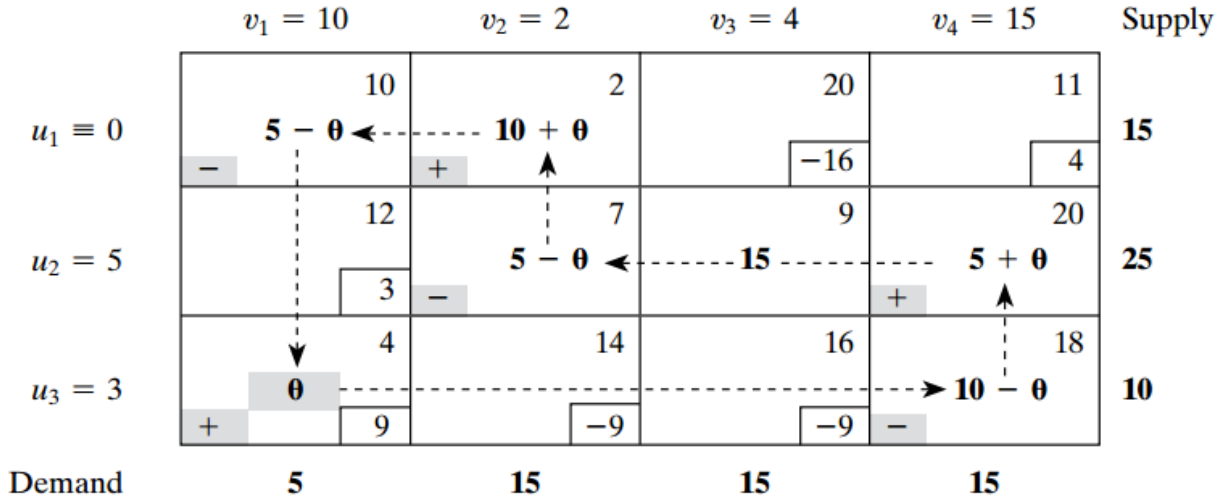
Basic	x_{11}	x_{12}	x_{13}	x_{14}	x_{21}	x_{22}	x_{23}	x_{24}	↓ x_{31}	x_{32}	x_{33}	x_{34}
Z	0	0	−16	4	3	0	0	0	9	−9	−9	0

Because the transportation model *minimizes* cost, the entering variable is the one having the *most positive* coefficient in the z-row—namely, x_{31} is the entering variable.

STEP 3: Determine the Leaving Variable

- Construct a *closed loop* that starts and ends at the entering variable cell (3, 1). The loop consists of *connected* horizontal and vertical segments only (no diagonals are allowed).
- Alternate between subtracting (-) and adding (+) the amount θ at the successive *corners* of the loop.
- Determine the maximum value of θ

.



For $\theta \geq 0$, the new values of all the variables remain nonnegative if

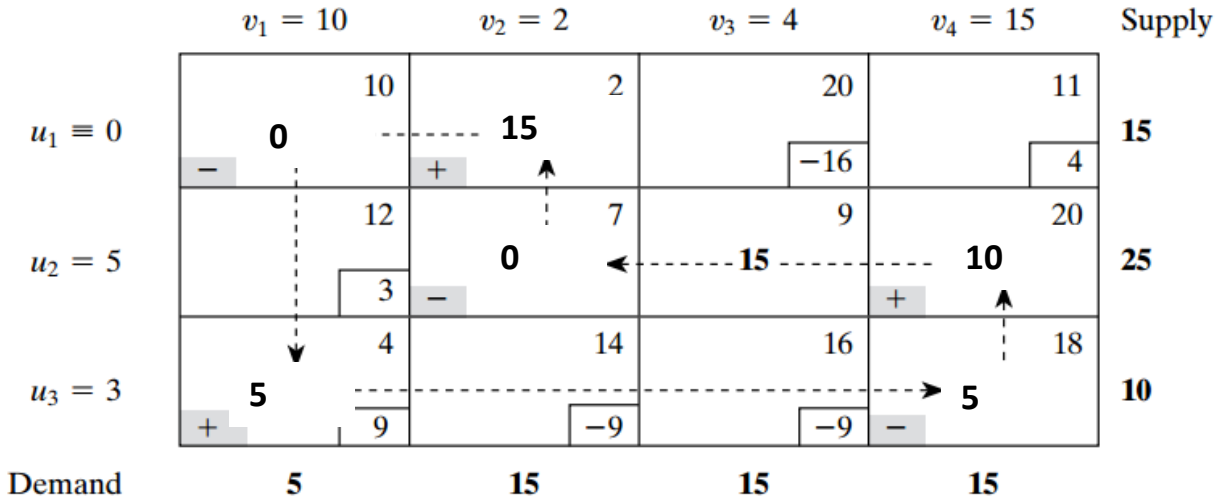
$$x_{11} = 5 - \theta \geq 0$$

$$x_{22} = 5 - \theta \geq 0$$

$$x_{34} = 10 - \theta \geq 0$$

STEP 3: Determine the Leaving Variable (cont'd)

- The corresponding maximum value of θ is 5, which occurs when both x_{11} and x_{22} reach zero level. (Select the variable with the higher unit cost). Hence choose x_{11} with $c_{11} = 10$ as opposed to $c_{22} = 7$, as the leaving variable.
- The values of the basic variables at the corners of the closed loop are adjusted to accommodate setting $x_{31} = 5$,



STEP 2 (Iteration 2): Determine the Entering Basic Variable (cont'd)

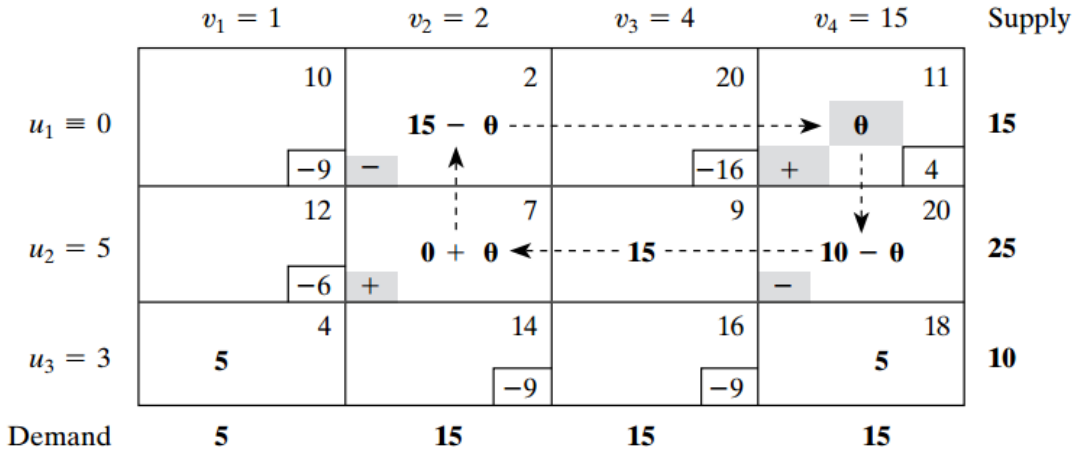
- Given the new basic solution, repeat the computation of the multipliers *u* and v and evaluate the non-basic variables .

	$v_1 = 1$	$v_2 = 2$	$v_3 = 4$	$v_4 = 15$	
$u_1 \equiv 0$	10 [−9]	15 2	20 [−16]	11 [4]	15
$u_2 = 5$	12 [−6]	0 7	15 9	10 20	25
$u_3 = 3$	5 4	14 [−9]	16 [−9]	5 18	10
	5	15	15	15	

*u*1 + v2 = 2; u1=0, v2=2
*u*2 + v2 = 7; u2=5, v2=2
*u*2 + v3 =9; u2=5, v3=4
*u*2 + v4 = 20; u2=5, v4=15
*u*3 + v4 = 18; u3=3, v4=15
*u*3 + v1 = 4; u3=3, v1=1

- The entering variable is *x*14 (*most positive* coefficient in the *z*-row).

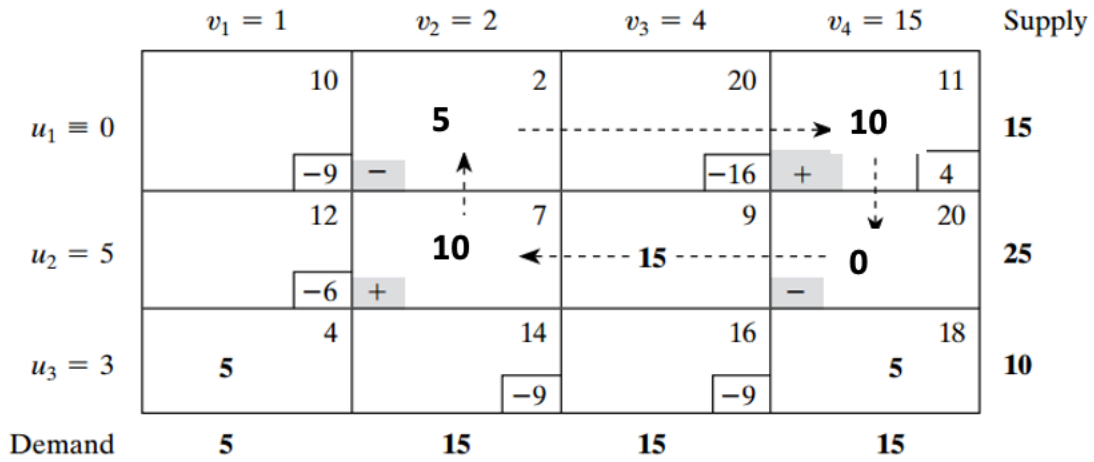
STEP 2 (Iteration 2): Determine the Entering Basic Variable (cont'd)



- Construct a *closed loop* that starts and ends at the entering variable cell (1, 4). The corresponding maximum value of θ is 10.
- The leaving variable is x_{24}.

STEP 2 (Iteration 2): Determine the Entering Basic Variable (cont'd)

- The values of the basic variables at the corners of the closed loop are adjusted to accommodate setting $x_{14} = 10$,



STEP 2 (Iteration 3): Optimal Solution

- Given the new basic solution, repeat the computation of the multipliers u and v.

	$v_1 = -3$	$v_2 = 2$	$v_3 = 4$	$v_4 = 11$	Supply
$u_1 \equiv 0$	10 **5** −13	2	20 −16	11 **10**	**15**
$u_2 = 5$	12 −10	7 **10**	9 **15**	20 −4	**25**
$u_3 = 7$	4 **5**	14 −5	16 −5	18 **5**	**10**
Demand	**5**	**15**	**15**	**15**	

$u1 + v2 = 2$; u1=0, v2=2
$u2 + v2 = 7$; u2=5, v2=2
$u2 + v3 = 9$; u2=5, v3=4
$u1 + v4 = 11$; u1=0, v4=11
$u3 + v4 = 18$; u3=7, v4=11
$u3 + v1 = 4$; u3=7, v1=-3

- The new values of $u_i + v_j - c_{ij}$ are now negative for all non-basic x_{ij}. Thus, the solution is optimal.

From silo	To mill	Number of truckloads
1	2	5
1	4	10
2	2	10
2	3	15
3	1	5
3	4	5
Optimal cost = $435		

The Transportation Model

Balanced Transportation Problem

Model Formulation =>

$$\text{Minimize} \quad Z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij},$$

subject to

$$\sum_{j=1}^{n} x_{ij} = s_i \qquad \text{for } i = 1, 2, \ldots, m,$$

$$\sum_{i=1}^{m} x_{ij} = d_j \qquad \text{for } j = 1, 2, \ldots, n,$$

and

$$x_{ij} \geq 0, \qquad \text{for all } i \text{ and } j.$$

x_{ij} for all i and j is said to be balanced transportation problem when total supply from all the sources is equal to the total demand in all destinations. Otherwise, problem is said to be unbalanced transportation problem.

The Transportation Model

Unbalanced Transportation Problem

A transportation problem is said to be **unbalanced** if the supply and demand are not equal.

Two situations are possible:-

1. If Supply < demand, a dummy supply variable is introduced in the equation to make it equal to demand.
2. If demand < supply, a dummy demand variable is introduced in the equation to make it equal to supply.

The unit transportation cost for the dummy supply variable and dummy demand variable are assigned zero values, because no shipment is made.

Example 1 – Balanced TM

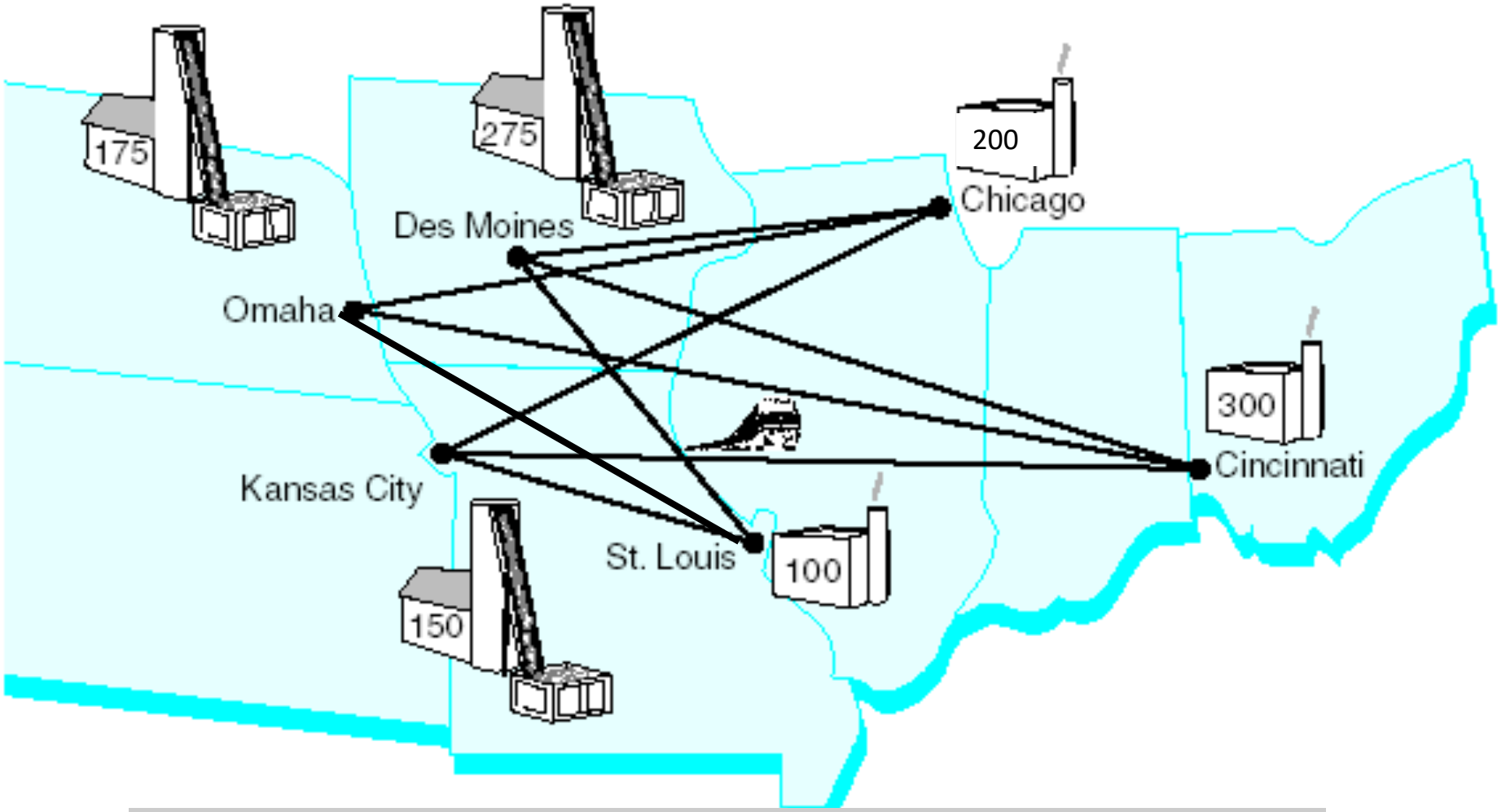
How many tons of wheat to transport from each grain elevator to each mill on a monthly basis in order to minimize the total cost of transportation?

Grain Elevator	Supply	Mill	Demand
1. Kansas City	150	A. Chicago	200
2. Omaha	175	B. St. Louis	100
3. Des Moines	275	C. Cincinnati	300
Total	600 tons	Total	600 tons

Transport Cost from Grain Elevator to Mill ($/ton)

Grain Elevator	A. Chicago	B. St. Louis	C. Cincinnati
1. Kansas City	$ 6	$ 8	$ 10
2. Omaha	7	11	11
3. Des Moines	4	5	12

Example 1 (cont'd)



Network of Transportation Routes for Wheat Shipments

Example 1 (cont'd)

Model Formulation

Transport Cost from Grain Elevator to Mill ($/ton)			
Grain Elevator	**A. Chicago**	**B. St. Louis**	**C. Cincinnati**
1. Kansas City	**$ 6**	**$ 8**	**$ 10**
2. Omaha	**7**	**11**	**11**
3. Des Moines	**4**	**5**	**12**

Grain Elevator	Supply	Mill	Demand
1. Kansas City	150	A. Chicago	200
2. Omaha	175	B. St. Louis	100
3. Des Moines	275	C. Cincinnati	300
Total	600 tons	Total	600 tons

Minimize $Z = 6x_{1A} + 8x_{1B} + 10x_{1C} + 7x_{2A} + 11x_{2B} + 11x_{2C} + 4x_{3A} + 5x_{3B} + 12x_{3C}$

subject to:

$x_{1A} + x_{1B} + x_{1C} = 150$ (Supply from Kansas City)

$x_{2A} + x_{2B} + x_{2C} = 175$ (Supply from Kansas Omaha)

$x_{3A} + x_{3B} + x_{3C} = 275$ (Supply from Kansas Des Moines)

$x_{1A} + x_{2A} + x_{3A} = 200$ (Demand at St. Louis)

$x_{1B} + x_{2B} + x_{3B} = 100$ (Demand at Cincinnati)

$x_{1C} + x_{2C} + x_{3C} = 300$

$x_{mn} \geq 0$

x_{mn} = tons of wheat from each grain elevator, m, m = 1, 2, 3, to each mill n, n = A,B,C

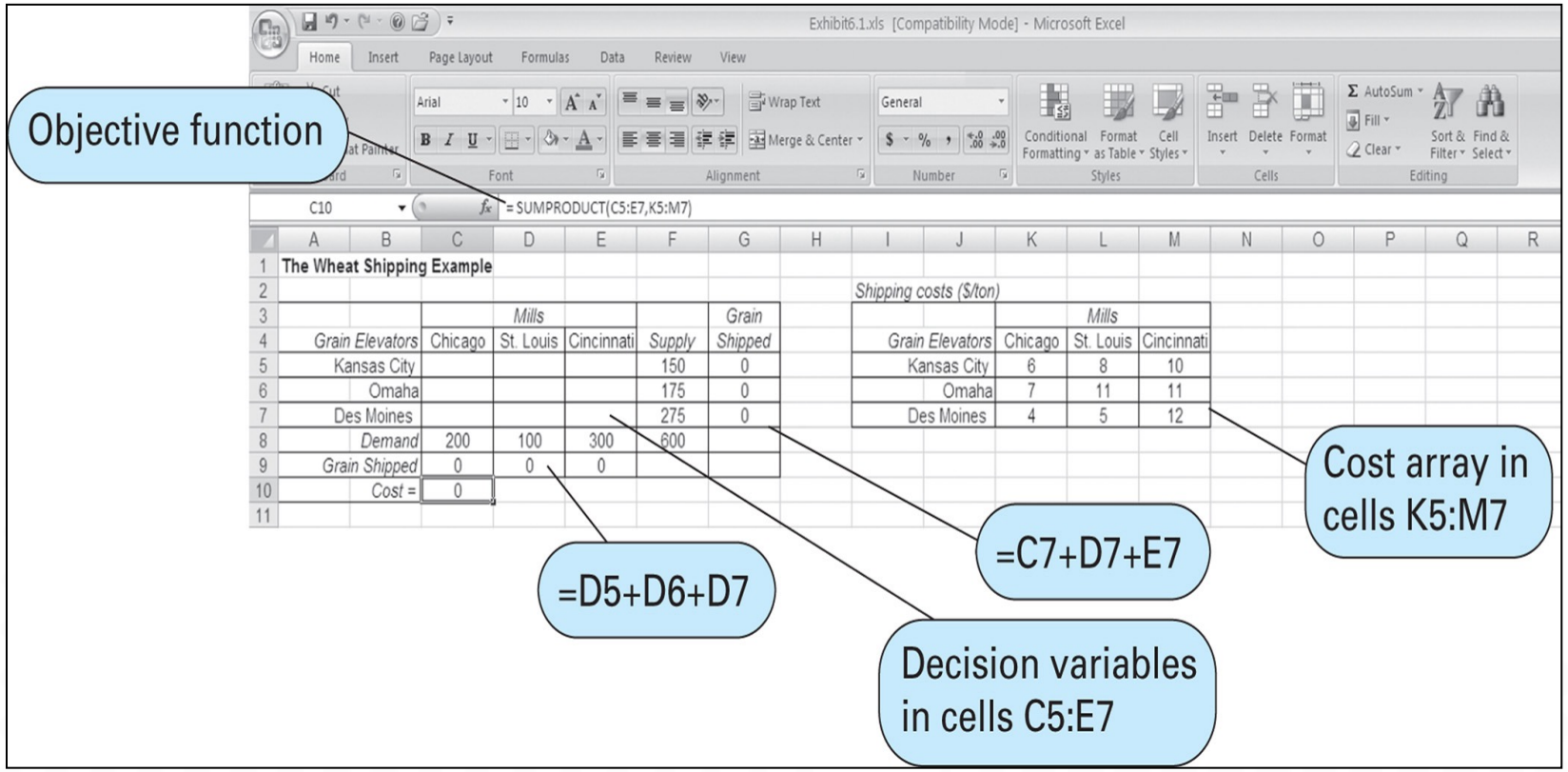
Example 1 (cont'd)

Transportation Model Solution Using Excel Solver

Grain Elevator	Supply	Mill	Demand
1. Kansas City	150	A. Chicago	200
2. Omaha	175	B. St. Louis	100
3. Des Moines	275	C. Cincinnati	300
Total	600 tons	Total	600 tons

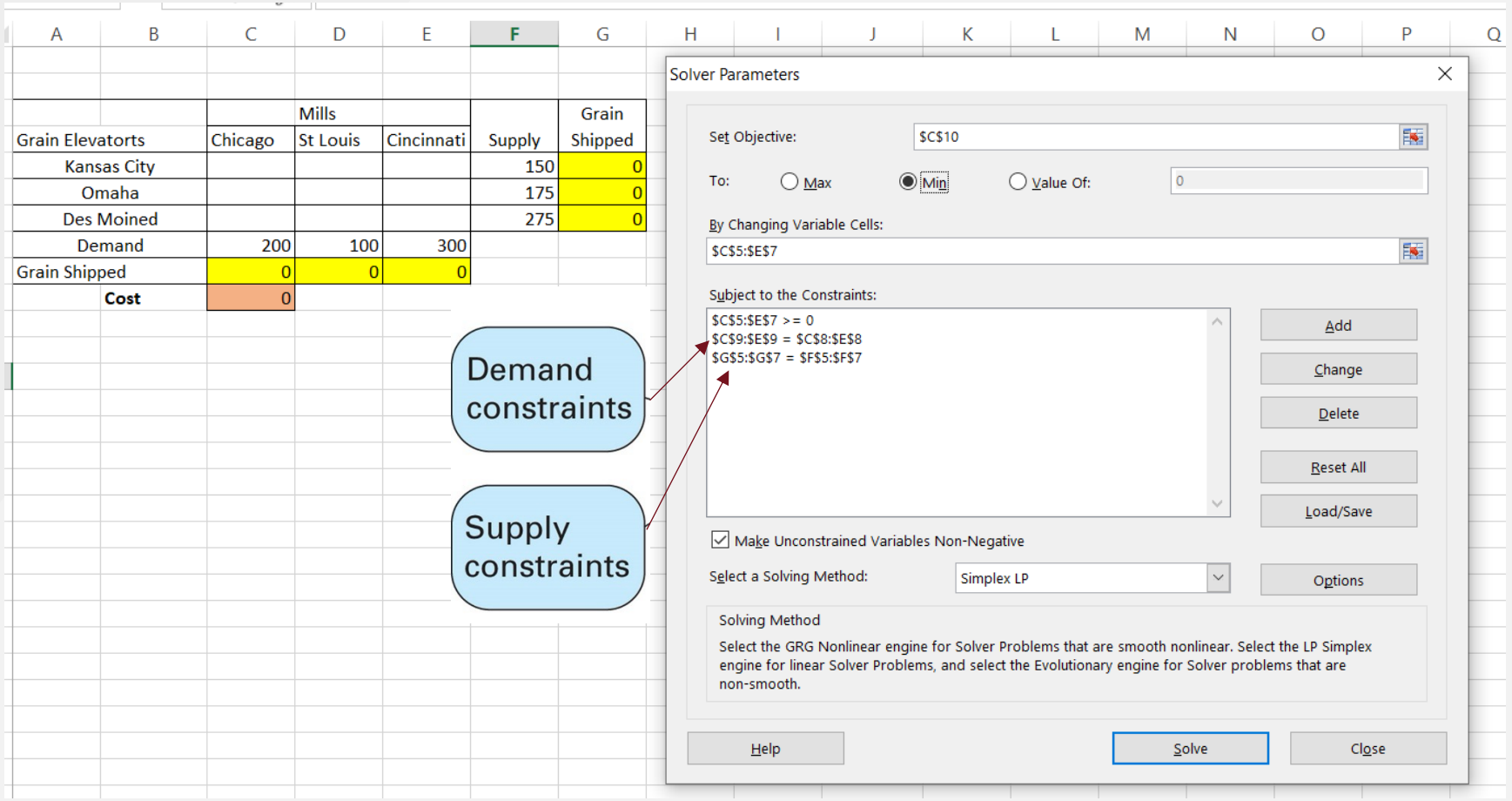
Transport Cost from Grain Elevator to Mill ($/ton)

Grain Elevator	A. Chicago	B. St. Louis	C. Cincinnati
1. Kansas City	$ 6	$ 8	$ 10
2. Omaha	7	11	11
3. Des Moines	4	5	12



Example 1 (cont'd)

Transportation Model Solution Using Excel Solver



		Mills			Grain
Grain Elevatorts	Chicago	St Louis	Cincinnati	Supply	Shipped
Kansas City				150	0
Omaha				175	0
Des Moined				275	0
Demand	200	100	300		
Grain Shipped	0	0	0		
Cost	0				

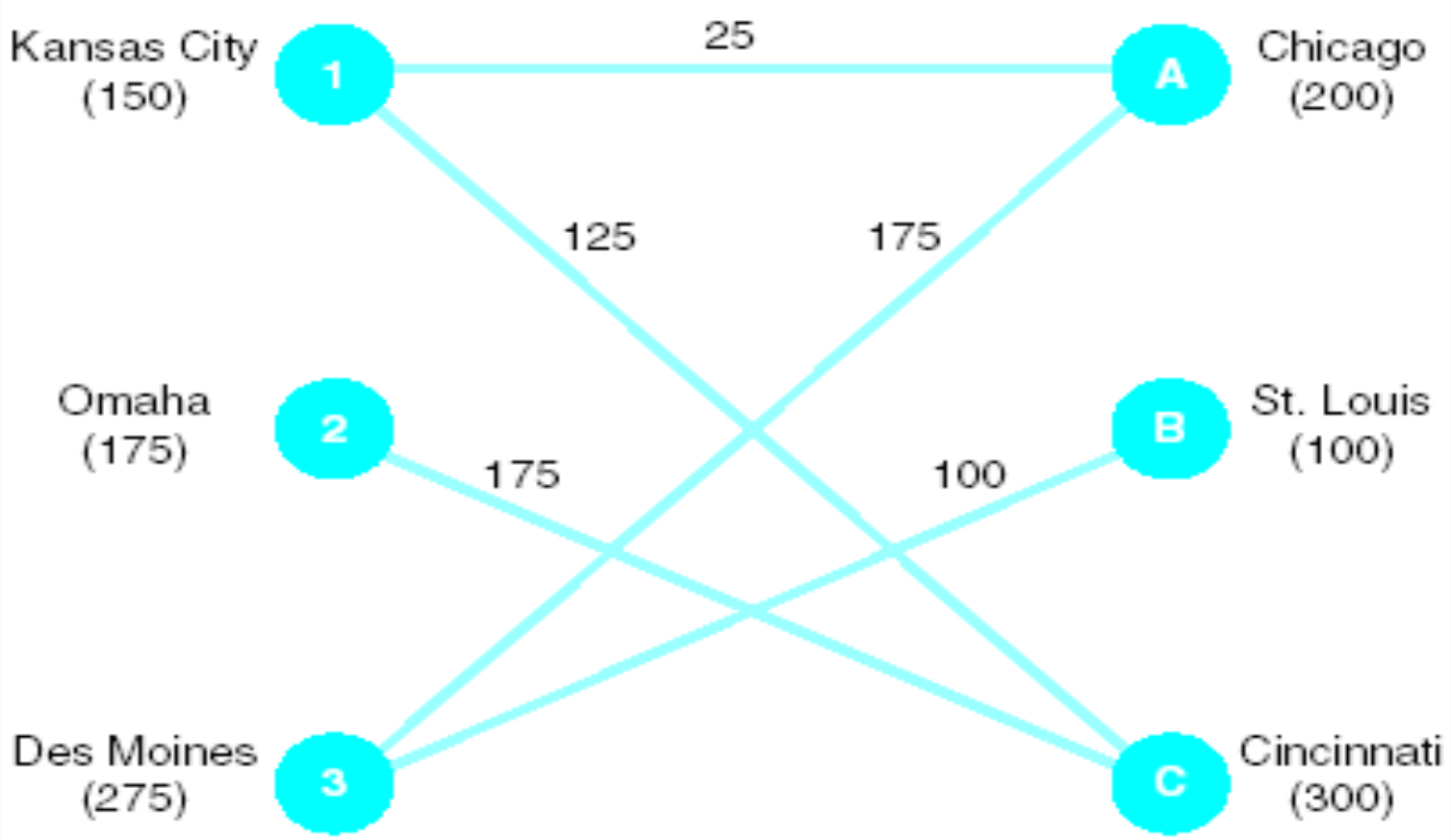
Example 1 (cont'd)

Transportation Model Solution Using Excel Solver

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	**The Wheet Shipping Example**												
2									Shiping Costs				
3				Mills			Grain					Mills	
4	Grain Elevatorts		Chicago	St Louis	Cincinnati	Supply	Shipped		Grain Elevatorts		Chicago	St Louis	Cincinnati
5		Kansas City	25	0	125	150	150			Kansas City	6	8	10
6		Omaha	0	0	175	175	175			Omaha	7	11	11
7		Des Moined	175	100	0	275	275			Des Moined	4	5	12
8		Demand	200	100	300								
9	Grain Shipped		200	100	300								
10		**Cost**	4525										

Example 1 (cont'd)

Transportation Model Solution (summary)



Example 1 (cont'd)

Transportation Model Solution Using Python

1. Import library (PuLP) and define node and capacities

```
#Import PuLP modeler functions
from pulp import *

#Creates a list of all the supply nodes
GrainE = ["1", "2","3"]

#Creates a dictionary for the number of units of supply for each supply node
supply = {"1": 150,
          "2": 175,
          "3": 275}
#Creates a list of all demand nodes
Mills = ["A", "B", "C"]
#Creates a dictionary for the number of units of demand for each demand node
demand = {"A":200,
          "B":100,
          "C":300 }
```

Example 1 (cont'd)

Transportation Model Solution Using Python (cont'd)

2. Define cost shipping
The cost data is then inputted into a list, containing the costs of shipping from the supply to demand nodes. Note here that the commenting and structure of the code makes the data appear as a table for easy editing. The *Grain Elevator* and *Mills* lists (Supply and Demand nodes) are added to make a large list (of all nodes) and inputted into PuLPs *makeDict* function.

```
costs = [#Grain Elevator
        #A B C
        [6,8,10],# 1   Mills
        [7,11,11], #2
        [4,5,12] #3
        ]

# The cost data is made into a dictionary
costs = makeDict([GrainE,Mills],costs,0)
```

Example 1 (cont'd)

Transportation Model Solution Using Python (cont'd)

3. Create the variable to contain the problem data.

   Install using LpProblem function.

```
# Creates the 'prob' variable to contain the problem data
prob = LpProblem("Grain Distribution Problem",LpMinimize)
```

It has two parameters,

I. The arbitrary name of this problem (as a string),

II. Second parameter being either LPMinimize or LpMaximize depending on the type of LP to solve.

Example 1 (cont'd)

Transportation Model Solution Using Python (cont'd)

4. Create **Routes**.

   A list of tuples is created containing all the possible route for the transportation.

```
# Creates a list of tuples containing all the possible routes for transport
Routes = [(i,j) for i in GrainE for j in Mills]
```

5. Create **vars** dictionary.

   A dictionary called *vars* is created which contains the LP variables. The reference keys to the dictionary are the Grain Elevator name, then the mills name(*["1"]["A"]*) , and the data is *Route_Tuple*. (e.g. *["1"]["A"]*: Route_1_A). The lower limit of zero is set, the upper limit of *None* is set, and the variables are defined to be Integers.

```
# A dictionary called 'vars' is created to contain the referenced variables(the routes)
vars = LpVariable.dicts("Route",(GrainE,Mills),0,None,LpInteger)
```

Example 1 (cont'd)

Transportation Model Solution Using Python (cont'd)

6. Set the objective function.

   The objective function is added to the variable *prob* using a list comprehension. Since *vars* and *costs* are now dictionaries (with further internal dictionaries), they can be used as if they were tables, as *for (i,j) in Routes* will cycle through all the combinations/arcs.

```
# The objective function is added to 'prob' first
prob += lpSum([vars[i][j]*costs[i][j] for (i,j) in Routes]), "Sum_of_Transporting_Costs"
```

Example 1 (cont'd)

Transportation Model Solution Using Python (cont'd)

7. Set the supply constraints.
   - The supply constraints are added using a normal *for* loop and a list comprehension.
   - Supply Constraints: For each Grain Elevator in turn, the values of the decision variables (number of grain cases on arc) to each of the mills is summed, and then constrained to being less than or equal to the supply max for that grain elevator.
   - "Add constraints '+=' operator again to add more data to the variable prob.

```
# The supply maximum constraints are added to prob for each supply node (Grain Elavator)
for i in GrainE:
    prob += lpSum([vars[i][j] for j in Mills]) ==supply[i], "Sum_of_Products_out_of_GrainE_%s"%i
```

Example 1 (cont'd)

Transportation Model Solution Using Python (cont'd)

8. Set the demand constraints.
   - The demand constraints are added using a normal *for* loop and a list comprehension.
   - Demand Constraints: For each mill in turn, the values of the decision variables (number on arc) from each of the grain Elavator is summed, and then constrained to being greater than or equal to the demand minimum.
   - "Add constraints '+=' operator again to add more data to the variable prob.

```
# The demand minimum constraints are added to prob for each demand node (Mills)
for j in Mills:
    prob += lpSum([vars[i][j] for i in GrainE]) ==demand[j], "Sum_of_Products_into_Mill%s"%j
```

Example 1 (cont'd)

Transportation Model Solution Using Python (cont'd)

9. Add **WriteLP** function
   - WriteLP function can be used to copy the information into a .lp file into the directory that the code-block is running from.
   - There is no assignment operator (such as an equal's sign) on this line. This is because the function/method WriteLP called is being performed to the variable/object prob.
   - The dot . between the variable/object and the function/method is important and is seen frequently in Object Oriented software

```
# The problem data is written to an .lp file
prob.writeLP("GrainDistributionProblem.lp")
```

Example 1 (cont'd)

Transportation Model Solution Using Python (cont'd)

10. Add **solve()** function
    - The LP is solved using the solver that PuLP chooses. The input brackets after solve( )are left empty in this case, however they can be used to specify which solver to use.

```
# The problem is solved using PuLP's choice of Solver
prob.solve()
```

Example 1 (cont'd)

Transportation Model Solution Using Python (cont'd)

11 . Display output

- Now the results of the solver call can be displayed as output.
- Request the status of the solution, which can be one of "Not Solved", "Infeasible", "Unbounded", "Undefined" or "Optimal".
- The value of prob is returned as an integer, which must be converted to its significant text meaning using the lpStatus dictionary.

```
# The status of the solution is printed to the screen
print ("Status:", LpStatus[prob.status])
```

Example 1 (cont'd)

Transportation Model Solution Using Python (cont'd)

11 . Display output (cont'd)

- The variables and their resolved optimum values can now be printed to the screen.
- The for loop is used to called all the problem variable names. Prints each variable name, followed by an equal's sign, followed by its optimum value. The optimised objective function value is printed to the screen, using the value (prob.objective) function

```
# Each of the variables is printed with it's resolved optimum value
for v in prob.variables():
    print(v.name, "=", v.varValue)
# The optimised objective function value is printed to the screen
print ("Total Cost of Transportation = ", value(prob.objective))
```

```
Status: Optimal
Route_1_A = 25.0
Route_1_B = 0.0
Route_1_C = 125.0
Route_2_A = 0.0
Route_2_B = 0.0
Route_2_C = 175.0
Route_3_A = 175.0
Route_3_B = 100.0
Route_3_C = 0.0
Total Cost of Transportation =  4525.0
```

Example 2 – Unbalanced TM

A Cola Company has two warehouses from which it distributes cola to five carefully chosen stores. At the start of every week, each stores sends an order to the Cola Company head office, the Cola is then dispatched from the appropriate warehouse to the store. The Cola company would like to have an interactive computer program which they can run week by week to tell them which warehouse should supply which store so as to minimize the costs of the whole operation.

Suppose that at the start of a given week the company has 1000 cases of Cola cans at warehouse A, and 4000 cases of Cola cans at warehouse B, and that the stores(1,2,3,4,5) require 500, 900, 1800, 200, and 700 cases, respectively. Which warehouse should supply which store?

Solve it using Excel Solver and Python.

Example 2 - Unbalanced

Cost for warehouse A and B to Store (1,2,3,4,5)

From warehouse to store	A	B
1	2	3
2	4	1
3	5	3
4	2	2
5	1	3

Example 2 - Solution

This example for unbalanced Transportation model.

Solution Using Excel Solver :

The Wheat Shipping Example

		Warehouse			*Cola*
	Store	A	B	**Demand**	*Shipped*
	1	300	200	500	500
	2	0	900	900	900
	3	0	1800	1800	1800
	4	0	200	200	200
	5	700	0	700	700
	D	0	900	900	900
	Supply	1000	4000		
	Cola Shipped	1000	4000		
Obj. fnc:	**COST =**	$ 8,600			

Shipping Costs ($/box)

		Warehouse	
	Store	A	B
	1	2	3
	2	4	1
	3	5	3
	4	2	2
	5	1	3
	D	0	0

Example 2 - Solution

Solution Using Python:

```
# Creates a dictionary for the number of units of supply for each supply node
supply = {"A": 1000,
          "B": 4000}

# Creates a list of all demand nodes
Stores = ["1", "2", "3", "4", "5","D"]

# Creates a dictionary for the number of units of demand for each demand node
demand = {"1":500,
          "2":900,
          "3":1800,
          "4":200,
          "5":700,
          "D":900}
costs = [#Store
         #1 2 3 4 5 D
         [2,4,5,2,1,0 ],#A   Warehouses
         [3,1,3,2,3,0 ] #B
         ]
```

Example 2 - Solution

Solution Using Python (cont'd)

```
# The supply maximum constraints are added to prob for each supply node (warehouse)
for w in Warehouses:
    prob += lpSum([vars[w][b] for b in Stores])==supply[w], "Sum_of_Products_out_of_Warehouse_%s"%w

# The demand minimum constraints are added to prob for each demand node (bar)
for b in Stores:
    prob += lpSum([vars[w][b] for w in Warehouses])==demand[b], "Sum_of_Products_into_Stores%s"%b
# The problem data is written to an .lp file
```

```
Status: Optimal
Route_A_1 = 300.0
Route_A_2 = 0.0
Route_A_3 = 0.0
Route_A_4 = 0.0
Route_A_5 = 700.0
Route_A_D = 0.0
Route_B_1 = 200.0
Route_B_2 = 900.0
Route_B_3 = 1800.0
Route_B_4 = 200.0
Route_B_5 = 0.0
Route_B_D = 900.0
Total Cost of Transportation =  8600.0
```

Exercise 1

Consider the transportation models in Table 1. Compare the starting solutions obtained by the northwest-corner, least-cost, and Vogel methods. Then, validate the optimal solution obtain using Excel Solver.

Table 1

To: / From:	Project #1	Project #2	Project #3	Supply
Farm A	4	2	8	100
Farm B	5	1	9	200
Farm C	7	6	3	200
Demand	50	150	300	