

1 The Transportation Problem

1.1 An LP Formulation

Suppose a company has m warehouses and n retail outlets. A single product is to be shipped from the warehouses to the outlets. Each warehouse has a given level of supply, and each outlet has a given level of demand. We are also given the transportation costs between every pair of warehouse and outlet, and these costs are assumed to be linear. More explicitly, the assumptions are:

- The total supply of the product from warehouse i is a_i , where $i = 1, 2, \dots, m$.
- The total demand for the product at outlet j is b_j , where $j = 1, 2, \dots, n$.
- The cost of sending one unit of the product from warehouse i to outlet j is equal to c_{ij} , where $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$. The total cost of a shipment is linear in the size of the shipment.

The problem of interest is to determine an optimal transportation scheme between the warehouses and the outlets, subject to the specified supply and demand constraints. Graphically, a transportation problem is often visualized as a network with m source nodes, n sink nodes, and a set of $m \cdot n$ “directed arcs.” This is depicted in Figure 1. We now proceed with a linear-programming formulation of this problem.

1.2 The Decision Variables

A transportation scheme is a complete specification of how many units of the product should be shipped from each warehouse to each outlet. Therefore, the decision variables are:

x_{ij} = the size of the shipment from warehouse i to outlet j , where $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$.

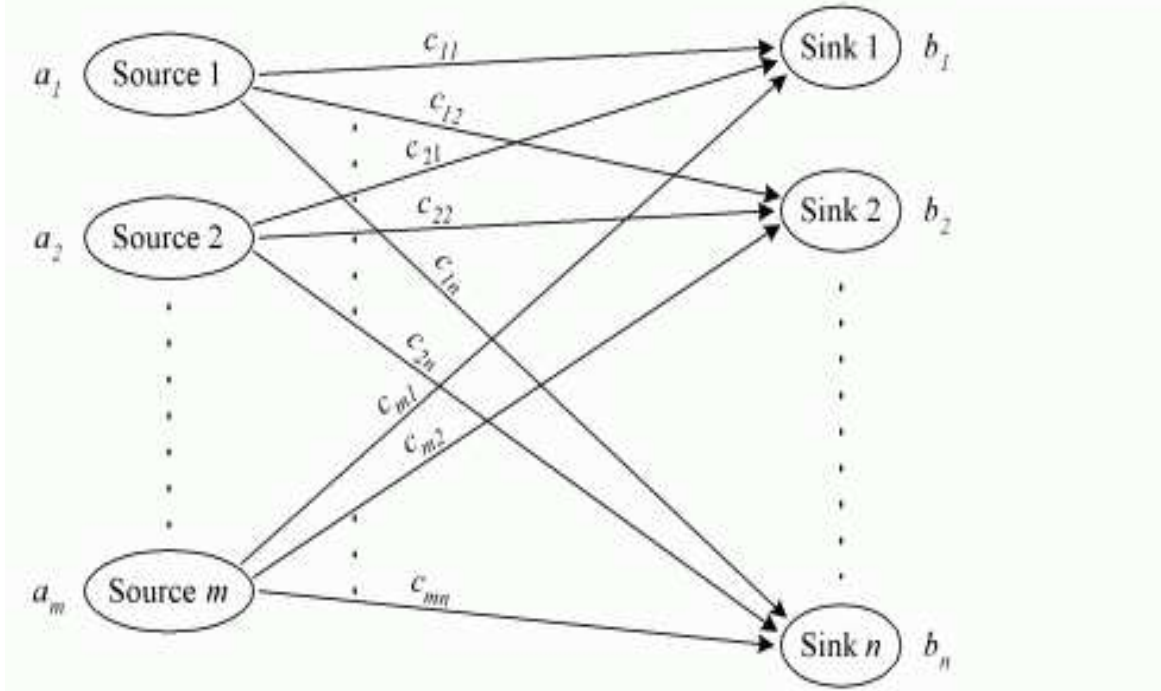
This is a set of $m \cdot n$ variables.

1.3 The Objective Function

Consider the shipment from warehouse i to outlet j . For any i and any j , the transportation cost per unit is c_{ij} ; and the size of the shipment is x_{ij} . Since we assume that the cost function is linear, the total cost of this shipment is given by $c_{ij}x_{ij}$. Summing over all i and all j now yields the overall transportation cost for all warehouse-outlet combinations. That is, our objective function is:

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

Figure TP-1

Figure 1: A transportation network with m sources and n sinks.

1.4 The Constraints

Consider warehouse i . The total outgoing shipment from this warehouse is the sum $x_{i1} + x_{i2} + \cdots + x_{in}$. In summation notation, this is written as $\sum_{j=1}^n x_{ij}$. Since the total supply from warehouse i is a_i , the total outgoing shipment cannot exceed a_i . That is, we must require

$$\sum_{j=1}^n x_{ij} \leq a_i, \quad \text{for } i = 1, 2, \dots, m.$$

Consider outlet j . The total incoming shipment at this outlet is the sum $x_{1j} + x_{2j} + \cdots + x_{mj}$. In summation notation, this is written as $\sum_{i=1}^m x_{ij}$. Since the demand at outlet j is b_j , the total incoming shipment should not be less than b_j . That is, we must require

$$\sum_{i=1}^m x_{ij} \geq b_j, \quad \text{for } j = 1, 2, \dots, n.$$

This results in a set of $m + n$ functional constraints. Of course, as physical shipments, the x_{ij} 's should be nonnegative.

1.5 LP Formulation

In summary, we have arrived at the following formulation:

$$\begin{aligned}
 &\text{Minimize} && \sum_{i=1}^m \sum_{j=1}^n c_{ij}x_{ij} \\
 &\text{Subject to:} && \\
 &&& \sum_{j=1}^n x_{ij} \leq a_i \quad \text{for } i = 1, 2, \dots, m \\
 &&& \sum_{i=1}^m x_{ij} \geq b_j \quad \text{for } j = 1, 2, \dots, n \\
 &&& x_{ij} \geq 0 \quad \text{for } i = 1, 2, \dots, m \text{ and } j = 1, 2, \dots, n.
 \end{aligned}$$

This is a linear program with $m \cdot n$ decision variables, $m + n$ functional constraints, and $m \cdot n$ nonnegativity constraints.

1.6 A Numerical Example

In an actual instance of the transportation problem, we need to specify m and n , and replace the a_i 's, the b_j 's, and the c_{ij} 's with explicit numerical values. As a simple example, suppose we are given: $m = 3$ and $n = 2$; $a_1 = 45$, $a_2 = 60$, and $a_3 = 35$; $b_1 = 50$ and $b_2 = 60$; and finally, $c_{11} = 3$, $c_{12} = 2$, $c_{21} = 1$, $c_{22} = 5$, $c_{31} = 2$, and $c_{32} = 4$. Then, substitution of these values into the above formulation leads to the following explicit problem:

$$\begin{aligned}
 &\text{Minimize} && 3x_{11} & +2x_{12} & +x_{21} & +5x_{22} & +2x_{31} & +4x_{32} \\
 &\text{Subject to:} && \\
 &&& x_{11} & +x_{12} & & & & & \leq 45 & (1) \\
 &&& & & x_{21} & +x_{22} & & & \leq 60 & (2) \\
 &&& & & & & x_{31} & +x_{32} & \leq 35 & (3) \\
 &&& x_{11} & & +x_{21} & & +x_{31} & & \geq 50 & (4) \\
 &&& & x_{12} & & +x_{22} & & +x_{32} & \geq 60 & (5) \\
 &&& x_{ij} \geq 0 & \text{ for } i = 1, 2, 3 \text{ and } j = 1, 2.
 \end{aligned}$$

For an example of an explicit transportation scheme, let $x_{11} = 20$, $x_{12} = 20$, $x_{21} = 20$, $x_{22} = 20$, $x_{31} = 10$, and $x_{32} = 20$. It is easily seen that this proposed solution satisfies all of the constraints, and hence it is feasible. In words, the solution calls for shipping 20 units from warehouse 1 to outlet 1, 20 units from warehouse 1 to outlet 2, 20 units from warehouse 2 to outlet 1, \dots , and finally 20 units from warehouse 3 to outlet 2. The total transportation cost associated with this transportation scheme can be computed as:

$$3 \cdot 20 + 2 \cdot 20 + 1 \cdot 20 + 5 \cdot 20 + 2 \cdot 10 + 4 \cdot 20 = 320.$$

1.7 An Equivalent Formulation

To derive an optimal transportation scheme for the above example, we can of course apply the standard Simplex algorithm. This means that we will first introduce 3 slack variables for constraints (1), (2), and (3) and 2 surplus variables for constraints (4) and (5), to convert these inequalities into equalities. On top of these, at the onset of Big-M, we also need to introduce 2 more artificial variables to serve as starting basic variables for constraints (4) and (5). We would then go through Big-M procedure. This routine is standard, but the introduction of so many new variables seems tedious. Can we do better? It turns out that we can.

The first observation is that for a given problem to have any feasible solution, the total supply must not be less than the total demand. In this numerical example, we have a total supply of 140 and a total demand of 110. Hence, feasible solutions exist; and indeed, we constructed one with ease. The second observation is that if the total supply happens to be equal to the total demand, then *any* feasible solution must satisfy all of the inequality constraints as equalities. (For example, this would be the case if a_1 , a_2 , and a_3 had been 40, 40, and 30, respectively.) As a consequence, whenever the given total supply and total demand are the same, we can replace all (functional) inequality constraints by equality constraints. Our third, and final, observation is that after such replacements, there is no longer any need for introducing slack or surplus variables.

Certainly, we cannot expect every problem to come with identical total supply and total demand. However, notice that if the total supply is strictly greater than the total demand, then one can artificially create a “dummy sink” to absorb the difference between the two. Of course, in order to preserve the original cost structure, the transportation cost for units sent to the dummy sink should be set to zero. Thus, for this specific example, we can introduce a third outlet to serve as the dummy sink; and let $b_3 = 30$ and $c_{13} = c_{23} = c_{33} = 0$. This yields the following new linear program:

$$\begin{array}{rllllllll}
 \text{Minimize} & 3x_{11} & +2x_{12} & & +x_{21} & +5x_{22} & & +2x_{31} & +4x_{32} & & \\
 \text{Subject to:} & & & & & & & & & & \\
 & x_{11} & +x_{12} & +x_{13} & & & & & & & = 45 \\
 & & & & x_{21} & +x_{22} & +x_{23} & & & & = 60 \\
 & & & & & & & x_{31} & +x_{32} & +x_{33} & = 35 \\
 & x_{11} & & & +x_{21} & & & +x_{31} & & & = 50 \\
 & & x_{12} & & & +x_{22} & & & +x_{32} & & = 60 \\
 & & & x_{13} & & & +x_{23} & & & +x_{33} & = 30 \\
 x_{ij} \geq 0 & \text{for } i = 1, 2, 3 \text{ and } j = 1, 2, 3.
 \end{array}$$

It should be clear that by construction, this problem is equivalent to the original one.

With the above discussion, we can now assume without loss of generality that *every* transportation problem comes with identical total supply and total demand. This gives rise to what is called the *standard form* of the transportation problem. Formally, under the assumption that

$$\sum_{i=1}^m a_i = \sum_{j=1}^n b_j$$

holds, the standard form of the transportation problem is:

$$\begin{array}{rll}
 \text{Minimize} & \sum_{i=1}^m \sum_{j=1}^n c_{ij}x_{ij} \\
 \text{Subject to:} & \sum_{j=1}^n x_{ij} = a_i & \text{for } i = 1, 2, \dots, m \\
 & \sum_{i=1}^m x_{ij} = b_j & \text{for } j = 1, 2, \dots, n \\
 & x_{ij} \geq 0 & \text{for } i = 1, 2, \dots, m \text{ and } j = 1, 2, \dots, n.
 \end{array}$$

Can we also do away with the need for introducing artificial variables? The answer again is a yes. However, we will delay this discussion a little bit.

1.8 The Transportation Tableau

The Simplex tableau serves as a very compact format for representing and manipulating linear programs. In the same spirit, we now introduce a tableau representation for transportation problems that are in the standard form.

For a problem with m sources and n sinks, the tableau will be a table with m rows and n columns. Specifically, each source will have a corresponding row; and each sink, a corresponding column. For ease of reference, we shall refer to the cell that is located at the intersection of the i th row and the j th column as “cell (i, j) ”. Parameters of the problem will be entered into various parts of the table in the format below.

			Sink j			
			⋮			
			⋮			
			⋮			
Source i	⋯	⋯	c_{ij}	x_{ij}	⋯	⋯
			⋮			a_i
			⋮			
			⋮			
				b_j		

That is, each row is labelled with its corresponding source name at the left margin; each column is labelled with its corresponding sink name at the top margin; the supply from source i is listed at the right margin of the i th row; the demand at sink j is listed at the bottom margin of the j th column; the transportation cost c_{ij} is listed in a subcell located at the upper-left corner of cell (i, j) ; and finally, the value of x_{ij} is to be entered at the lower-right corner of cell (i, j) .

Again, we will use the numerical example above to illustrate the just-described format. With the introduction of a dummy sink to balance the total supply and total demand, this problem has three sources and three sinks. Therefore, the parameters of the problem can be specified in the explicit 3·3 tableau below.

			Sinks			
			1	2	3	
			3	2	0	
1						45
			1	5	0	
Sources 2						60
			5	4	0	
3						35
			50	60	30	

Notice that we have left the spaces for the x_{ij} 's blank; these will be filled in later, when we begin the solution process. As a specific example, the feasible solution $x_{11} = 20$, $x_{12} = 20$, $x_{13} = 5$, $x_{21} = 20$,

$x_{22} = 20$, $x_{23} = 20$, $x_{31} = 10$, $x_{32} = 20$, and $x_{33} = 5$ would be entered as in the tableau below.

		Sinks			
		1	2	3	
Sources	1	3	2	0	45
		20	20	5	
	2	1	5	0	
	20	20	20		
3	5	4	0	35	
	10	20	5		
	50	60	30		

All transportation problems in the standard form will henceforth be specified in this compact tableau format.

1.9 Remarks

1. In most applications, the shipments are integer-valued. We have ignored this requirement in our formulation. However, it will turn out that if the specified a_i 's and b_j 's are integers, then the optimal solution of a transportation problem produced by the Simplex algorithm also is integer-valued.
2. In applications where an oversupply exists, it is often desirable to assume that there is an inventory-holding cost for units that are not shipped out. If this is the case, then we can easily accommodate the inventory-holding costs as part of the objective function by reinterpreting them as transportation costs to the dummy sink.
3. When the total supply is *less* than the total demand, it is still possible to create a balanced transportation problem. The idea is to create a dummy source and let it have a supply that equals the difference between the total demand and the total supply. In doing so, it may be reasonable to assess a shortage penalty cost for units that are sent (fictitiously) from the dummy source to any sink.

2 Constructing an Initial Basic Feasible Solution

We will use the previous numerical example to illustrate the methods. In algebraic form, our problem is:

$$\begin{array}{ll}
 \text{Minimize} & 3x_{11} + 2x_{12} + x_{21} + 5x_{22} + 2x_{31} + 4x_{32} \\
 \text{Subject to:} & \\
 & x_{11} + x_{12} + x_{13} = 45 \\
 & \phantom{x_{11} +} x_{21} + x_{22} + x_{23} = 60 \\
 & \phantom{x_{11} +} \phantom{x_{21} +} x_{31} + x_{32} + x_{33} = 35 \\
 & x_{11} + \phantom{x_{12} +} + x_{21} + \phantom{x_{22} +} + x_{31} = 50 \\
 & \phantom{x_{11} +} x_{12} + \phantom{x_{21} +} + x_{22} + \phantom{x_{23} +} + x_{32} = 60 \\
 & \phantom{x_{11} +} \phantom{x_{12} +} x_{13} + \phantom{x_{21} +} + x_{23} + \phantom{x_{23} +} + x_{33} = 30 \\
 & x_{ij} \geq 0 \quad \text{for } i = 1, 2, 3 \text{ and } j = 1, 2, 3;
 \end{array}$$

and in tableau form, this problem is specified as:

		Sinks			
		1	2	3	
Sources	1	3	2	0	45
	2	1	5	0	
	3	5	4	0	
		50	60	30	

An inspection of the algebraic form of the problem shows that we have a set of 6 equality constraints in 9 variables. We will first argue that it is possible to reduce the number of constraint equations by 1. This is a consequence of the following special feature of all *balanced* transportation problems. Notice that the three supply constraints sum up to

$$x_{11} + x_{12} + x_{13} + x_{21} + x_{22} + x_{23} + x_{31} + x_{32} + x_{33} = 140 ;$$

and that the three demand constraints also sum up to this same expression. It follows that it is sufficient to work with any subset of 5 out of these 6 equations. To understand what this means more explicitly, suppose we are given a proposed solution to the original equations, and we are asked to check whether or not the given solution is feasible. To answer this, we would have to substitute the proposed solution into the equations one by one. Now, observe that if the proposed solution is verified to satisfy the first 5 equations, then the equality of the sum of the supply equations and the sum of the demand equations implies that the proposed solution must also satisfy the last equation. Moreover, it should be clear that this observation holds regardless of the order in which the equations are checked. Thus, indeed, we can choose to remove any one of the given 6 equations, and doing so will not alter the feasible set in any way.

The next question of course is: Which equation should we remove? Interestingly, the answer is that we don't have to commit ourselves at this point. Intuitively, the reason behind this answer is that doing so would give us more flexibility; this will be made more apparent a little bit later.

With one equation removed (conceptually, that is), a basic feasible solution will have 5 basic variables and 4 nonbasic variables. If we are to solve this problem by the standard Simplex method, then our next task is to introduce 5 artificial variables (Actually, 4 is sufficient. Why?) and begin Phase I of the solution procedure. As noted earlier, it seems desirable to avoid introducing these additional variables (since we have to drive them out of the basis eventually). We are, therefore, motivated to explore other, hopefully more-direct, approaches. For this purpose, we will now switch to the simpler tableau representation of the problem.

Our first observation is that it is quite easy to construct a feasible solution to the problem. The general idea is to arbitrarily choose an empty cell in the above tableau and assign a value to the x_{ij} in that cell; after having done that, we then repeat the same process until every cell is assigned an explicit x_{ij} value. Now, as we begin this assignment process, notice that we do need to enforce the supply and demand constraints by making assignments that are such that: (i) the sum of the x_{ij} 's in every row equals the specified supply at the right margin of that row; and (ii) the sum of the x_{ij} 's in every column equals the specified demand at the bottom margin of that column. In other words, we are only interested in assignments with "correct"

row sums and column sums. As an example, our earlier solution, namely

		Sinks			
		1	2	3	
Sources	1	3	2	0	45
		20	20	5	
	2	1	5	0	
	20	20	20		
3	5	4	0	35	
	10	20	5		
	50	60	30		

satisfies these requirements; and in fact, it can be viewed as the outcome of such an assignment procedure. It should be clear now that this procedure is indeed very easy to implement; and you should attempt to construct a different feasible solution yourself.

Our goal, however, is to construct a basic feasible solution. This means that we actually have requirements that are stronger than just having correct row sums and column sums. Therefore, the question now is: What additional stipulations should be added to this procedure to ensure that the outcome is a basic feasible solution? This brings us to an important observation. Earlier, we made the point that out of a total of 9 variables, there are only 5 basic variables in every basic feasible solution. Since all nonbasic variables are assigned the value 0, a basic feasible solution must have at least 4 of its values equal to 0. (It is possible to have more than 4, since the basic solution may be degenerate.) It follows that we should avoid introducing “too many” positive entries into the tableau. For example, the feasible solution above does not have any 0 entry; therefore, it is not a basic solution.

An intuitive idea that seems to be helpful in this regard is that whenever we are about to assign a value into a cell, we should attempt to make that assignment as large as possible. Suppose the cell into which we are about to assign a value is cell (i, j) ; and let us refer to this cell as the *entering cell*. Notice that as a result of having possibly assigned previous entries into row i and into column j , part of the original supply from Source i has been depleted, and part of the original demand at Sink j has been met. Therefore, assigning the largest possible value into cell (i, j) means that this value should equal to either the *remaining supply* from Source i or the *remaining demand* at Sink j , whichever is smaller. It turns out that this additional stipulation is almost enough. Why “almost”? It is because we may go too far and end up with an insufficient number of basic variables.

Our next task, therefore, is to develop one more (and last) stipulation that is designed to eliminate the possibility of not having enough basic variables. For this purpose, we will now go through a careful examination of our assignment procedure.

2.1 Northwest Corner Rule

Suppose we begin the assignment process with cell $(1, 1)$ as the entering cell. Since this is our first attempt at an assignment, the remaining supply from Source 1 is 45, and the remaining demand at Sink 1 is 50. According to the above discussion, we should, therefore, assign 45 into cell $(1, 1)$ as the value of x_{11} . This immediately implies that x_{11} will serve as a basic variable. Moreover, since the remaining supply from Source 1 is exhausted as a result of this assignment, the key concept at this point is to also think of this action as designating x_{11} as the basic variable associated with equation (1). Since every equation is to have exactly one basic variable, we shall say that equation (1) has now received its “quota” of one basic variable. Having just filled its quota, we should therefore prevent equation (1) from “receiving” another basic variable.

Mechanically, this is achieved by crossing out the first row. Doing this will result in a new tableau with one less row, which will then serve as the starting point for a continuation of the same assignment procedure. Notice that to facilitate the process of making further assignments, we should now reduce the remaining demand at Sink 1 to 5, from 50. This completes what we shall refer to as an *assignment cycle*.

In summary, our first assignment cycle results in the revised tableau below.

		Sinks			
		1	2	3	
Sources	1*	3	2	0	45 → 0
		45			
	1	5	0		
	2				60
	5	4	0		35
3					
		50 → 5	60	30	

Thus, we have: (i) entered the value 45 into cell (1, 1) as x_{11} ; (ii) revised the original supply from Source 1 and the original demand at Sink 1 to a remaining supply of 0 and a remaining demand of 5, respectively; (iii) marked the first source (or row) with an “*” (asterisk) to indicate its removal.

The fact that exactly one row or one column is removed at the completion of an assignment cycle is critical, in that it is the additional stipulation that ensures that a correct number of basic variables is generated at the end of the entire assignment process. To comprehend this claim fully, let us now iterate our procedure to completion. Suppose cell (2, 1) is chosen as the next entering cell. Then, after assigning 5 as x_{21} , revising the remaining supply from Source 2 and the remaining demand at Sink 1 to 55 and 0 (respectively) and marking the removal of Sink 1, we obtain the new tableau below.

		Sinks			
		1*	2	3	
Sources	1*	3	2	0	45 → 0
		45			
	1	5	0		
	2				60 → 55
	5	4	0		35
3					
		50 → 5 → 0	60	30	

At this point, the remaining cells are (2, 2), (2, 3), (3, 2), and (3, 3). Suppose cell (2, 2) is chosen as the next entering cell. Then, after assigning 55 as x_{22} and going through another round of routine updates, we obtain

		Sinks			
		1*	2	3	
Sources	1*	3	2	0	45 → 0
		45			
	1	5	0		
	2*		55		60 → 55 → 0
	5	4	0		35
3					
		50 → 5 → 0	60 → 5	30	

and, at this point, only cells (3, 2) and (3, 3) remain. Suppose we choose to enter a 5 in cell (3, 2); then,

the new tableau is

		Sinks				
		1*	2*	3		
Sources	1*	3	45	2	0	45 → 0
	2*	1	5	55	0	60 → 55 → 0
	3	5	4	5	0	35 → 30
		50 → 5 → 0	60 → 5 → 0	30		

and we now have a single remaining cell, cell (3, 3). Notice that the remaining supply and the remaining demand are both at 30. Clearly, we should assign 30 as x_{33} ; and this assignment exhausts both the remaining supply and the remaining demand simultaneously. In a tied situation like this, we can choose to remove either Source 3 or Sink 3. Suppose Source 3 is chosen; then, a final round of updates leads to the tableau below.

		Sinks				
		1*	2*	3		
Sources	1*	3	45	2	0	45 → 0
	2*	1	5	55	0	60 → 55 → 0
	3*	5	4	5	0	35 → 30 → 0
		50 → 5 → 0	60 → 5 → 0	30 → 0		

This completes the entire assignment procedure, with the explicit assignments $x_{11} = 45$, $x_{21} = 5$, $x_{22} = 55$, $x_{32} = 5$, and $x_{33} = 30$. Cells without an explicit assignment are considered nonbasic; and therefore, their x_{ij} values are all equal to 0. The objective-function value of this solution is easily computed as:

$$3 \cdot 45 + 1 \cdot 5 + 5 \cdot 55 + 4 \cdot 5 + 0 \cdot 30 = 435.$$

That the final remaining supply and the final remaining demand are exhausted simultaneously is a direct consequence of the fact that we have a balanced transportation problem. It also confirms our earlier claim that out of the original 6 constraint equations, only 5 are necessary to define the feasible set.

More importantly, observe that after the removal of Source 3, which concludes the assignment process, Sink 3 remains as the only source or sink that has not been removed. (This is despite the fact that there is no more remaining cell.) Recall that the explicit assignment of an x_{ij} is tantamount to the designation of that x_{ij} as the basic variable associated with the constraint equation whose right-hand-side supply or demand is exhausted by this assignment. It follows that this observation is what allows us to assert that a correct number of basic variables is assigned at the end of this assignment procedure. In this example, this variable count equals 5, which comes from the total number of equations, $3 + 3$, minus the number of redundant equations, 1. In general, for a balanced transportation problem with m sources and n sinks, the correct basic-variable count is equal to $m + n - 1$.

Since equation (6) is the only equation that did not receive an assignment of an associated basic variable, we will consider it as the redundant equation. It should be clear that this identification is a result of the choice of our particular sequence of entering cells. For other choices, one could end up with the identification of a different equation as the redundant equation. A little bit of reflection now reveals that the fact that we did not declare at the outset of the solution procedure a specific equation as the redundant equation is

quite helpful, in the sense that doing so indeed offered more flexibility in our effort to construct an initial basic feasible solution.

Another related interesting observation is that if we had chosen cell (2, 2) as the first entering cell in the assignment process, then we would immediately encounter a tied situation. According to our tie-breaking strategy, we should assign 60 as x_{22} and remove either Source 2 or Sink 2 (but not both). Regardless of which of these is selected for removal, the remaining supply from the other source or the remaining demand at the other sink is also reduced to 0 at the same time. For the purpose of discussion, let us choose to remove Source 2; then, the fact that Sink 2, which has no remaining demand, continues to be available for further assignments implies that somewhere down the road in the remainder of the assignment process, we will be forced into assigning a 0 as the x_{ij} value in the second column. Such an *explicit* 0-assignment is important, in that it corresponds to the explicit declaration of the variable whose value is being assigned as a degenerate basic variable. Therefore, we should be sure not to miss, or skip, any such assignments. Otherwise, we would end up with an insufficient number of basic variables.

The assignment procedure described above is called the *northwest-corner method*. This name originates from the fact that at the start of each assignment cycle, the entering cell is always chosen to be the one that is located at the northwest corner of all remaining cells. Since this strategy makes no reference whatsoever to the c_{ij} values, it cannot, in general, be expected to produce good initial basic feasible solutions. (For example, the objective-function value of the above basic feasible solution equals 435, which is worse than that of the earlier feasible solution, at 320.) We will next describe two better procedures that take into account the fact that we are interested in minimizing $\sum_{i=1}^m \sum_{j=1}^n c_{ij}x_{ij}$.

2.2 The Least-Cost Method

The only difference between the least-cost method and the northwest-corner method is in the choice of entering variables. Here, the strategy is to always select the cell with the smallest c_{ij} value among all remaining cells as the entering cell. Ties are, as usual, broken arbitrarily.

Now, we will briefly go over an implementation of this new strategy. We will start with the initial tableau. Since all three c_{ij} 's in column 3 are equal to 0, we can choose any one of the cells in column 3 as the first entering cell. Let cell (1, 3) be our choice. Then, after assigning 30 to x_{13} and going through a round of updates, we obtain the tableau below.

		Sinks			
		1	2	3*	
	1	3	2	0	
1				30	45 → 15
	2	1	5	0	
2					60
	3	5	4	0	
3					35
		50	60	30 → 0	

The next entering cell is cell (2, 1). After assigning 50 to x_{21} and going through necessary updates, we

obtain the tableau below.

		Sinks			
		1*	2	3*	
Sources	1	3	2	0	30
	2	1	5	0	45 → 15
	3	5	4	0	60 → 10
		50 → 0	60	30 → 0	35

With only column 2 remaining, it is easily seen that we should then (sequentially) assign 15 as x_{12} , 35 as x_{32} , and finally 10 as x_{22} . This yields the final tableau below.

		Sinks			
		1*	2	3*	
Sources	1*	3	2	0	30
	2*	1	5	0	45 → 15 → 0
	3*	5	4	0	60 → 10 → 0
		50 → 0	60 → 45	30 → 0	35 → 0
		10 → 0			

Note that after assigning 10 to x_{22} , at the last step, we chose (arbitrarily) to remove Source 2, as opposed to removing Sink 2.

In conclusion, the assignments produced by the least-cost method are: $x_{12} = 15$, $x_{13} = 30$, $x_{21} = 50$, $x_{22} = 10$, and $x_{32} = 35$. This basic feasible solution has an objective-function value of 270, which is significantly better than the previous one.

2.3 The Vogel's Approximation Method

This section discusses another method to find an initial basic feasible solution but it is not central to further developments so it can be skipped without disturbing the continuity. We have seen that the least-cost method (typically) offers a significant improvement over the northwest-corner method. The method, however, may be construed as being myopic, in that the choice of the entering cell, at every iteration, is based solely on the location of the smallest available c_{ij} . The Vogel's approximation method will be slightly less myopic.

The strategy for the selection of entering cells in the Vogel's method is as follows. At the start of every assignment cycle, the first step is to compute (or update) a set of *penalties*, one for each row and one for each column, and then, in the second step, to select the entering cell on the basis of the relative magnitudes of these row and column penalties. Specifically, the penalty associated with a row or a column is defined to be the difference between the second-lowest cost and the lowest cost in that row or column; and the entering cell is chosen to be the cell with the smallest c_{ij} in the row or column with the greatest penalty. All ties are broken arbitrarily.

The intuitive basis behind this strategy is as follows. Suppose we wish to assign a value to an x_{ij} in a given row (column). If there are no existing preconditions, then the best possible choice, clearly, is to make an assignment into the least-cost cell in that row (column). However, in the course of an assignment process, we may have made earlier assignments into various parts of the given tableau; and as a result of these earlier assignments, we may not be able to assign anything into that least-cost cell. If this indeed happens to be the case, we would then be forced into an assignment in a higher-cost cell in that row (column). Now, observe that the incremental cost (rate) of such a "forced"

assignment is no less than the difference between the second-lowest cost and the least cost in the given row (column). It follows that the penalty associated with a row (column) can be interpreted as the incremental cost that would incur if we are unable to make an assignment into the least-cost cell in that row (column). With this interpretation, it should now be clear that the underlying intent of the strategy in the Vogel's method is to reduce, in every assignment cycle, the potential incremental costs associated with forced assignments.

Again, we will use the previous example to illustrate the method. It is easily seen that row 1 has a penalty of $2 - 0 = 2$, row 2 has a penalty of $1 - 0 = 1$, row 3 has a penalty of $4 - 0 = 4$, column 1 has a penalty of $3 - 1 = 2$, column 2 has a penalty of $4 - 2 = 2$, and finally column 3 has a penalty of $0 - 0 = 0$. The results of these calculations are displayed on the right and bottom margins of the tableau, as shown below.

		Sinks				
		1	2	3		
		3	2	0		
Sources	1				45	2
	2	1	5	0		
	3	5	4	0	60	1
					35	4
		50	60	30		
Penalty		2	2	0		

Since row 3 has the greatest penalty, the first entering cell will be in that row. Since cell (3, 3) is the least-cost cell in row 3, it will be the entering cell. After assigning 30 to x_{33} and going through a round of standard updates, we obtain the new tableau below.

		Sinks				
		1	2	3*		
		3	2	0		
Sources	1				45	2
	2	1	5	0		
	3	5	4	0	35 → 5	4
					50	60
		50	60	30 → 0		
Penalty		2	2	*		

(Notice that together with the removal of column 3, we have also crossed out the penalty associated with that column.) The fact that column 3 has been removed implies that the row penalties (but not the column penalties) should now be revised. Specifically, the penalty associated with row 1 should be revised to $3 - 2 = 1$; and similarly, those for row 2 and row 3 should be revised to $5 - 1 = 4$ and $5 - 4 = 1$, respectively. These revisions will be indicated on the right margin of the tableau, as shown below.

		Sinks				
		1	2	3*		
		3	2	0		
Sources	1				45	2 → 1
	2	1	5	0		
	3	5	4	0	35 → 5	4 → 1
					50	60
		50	60	30 → 0		
Penalty		2	2	*		

With these revisions, row 2 now has the greatest penalty; therefore, the next entering cell is cell (2, 1). After assigning

50 as x_{21} and updating in the usual manner, we obtain the tableau below.

		Sinks				
		1*	2	3*		
Sources	1	3	2	0	45	2 → 1
	2	1	5	0	60 → 10	1 → 4
	3	5	4	0	35 → 5	4 → 1
			50 → 0	60	30 → 0	
Penalty		*	2	*		

With only column 2 remaining, there is no need to update the penalties further. After assigning (sequentially) 45 as x_{12} , 5 as x_{32} , and 10 as x_{22} , we arrive at the final tableau below.

		Sinks				
		1*	2	3*		
Sources	1*	3	2	0	45 → 0	*
	2*	1	5	0	60 → 10 → 0	*
	3*	5	4	0	35 → 5 → 5	*
			50 → 0	60 → 15	30 → 0	
Penalty		*	2	*		

Note that after assigning 10 to x_{22} , at the last step, we chose (arbitrarily) to remove Source 2, as opposed to removing Sink 2.

In conclusion, the assignments produced by the Vogel's method are: $x_{12} = 45$, $x_{21} = 50$, $x_{22} = 10$, $x_{32} = 5$, and $x_{33} = 30$. This basic feasible solution has an objective-function value of 210, which happens to be an improvement over the one produced by the least-cost method. While it is true that solutions produced by the Vogel's method are typically better than those produced by the least-cost method, such an outcome cannot be guaranteed in general.

2.4 Remarks

1. Several other methods for constructing initial basic feasible solutions can be found in our text. These methods offer some differences in terms of total computational effort and in terms of the quality of the produced initial basic feasible solutions. In general, it is difficult to achieve a perfect balance between effort and quality. In fact, it may not even be desirable to do so, since constructing an initial basic feasible solution is only the first phase in the solution of a problem. In other words, it is the total solution effort for a problem that matters in the end. We will, therefore, not attempt to dwell upon a detailed discussion of these other methods.
2. In some applications, the objective may be to maximize the overall profit derived from the shipments. That is, in place of the c_{ij} 's, we could be given a set of p_{ij} 's, where p_{ij} is the profit per unit of shipment from Source i to Sink j . Clearly, the methods discussed above can be easily adapted to handle such problems. For example, the obvious counterpart of the least-cost method would select, in every assignment cycle, the cell with the greatest available p_{ij} as the entering cell. The Vogel's method can also be modified in a similar manner.

3 Progressing towards an Optimal Solution

After having constructed an initial basic feasible solution, our next task is to progress toward an optimal solution. Again, we will describe two methods for doing this. The first one is called the stepping-stone method; and the second, the u - v method.

3.1 The Stepping-Stone Method

Again, we will use the previous example to illustrate the method. The transportation tableau associated with the basic feasible solution produced by the least-cost method is given below.

		Sinks					
		1	2	3			
Sources	1	3	2	0			
						45	
						15	30
	2	1	5	0			
						60	
						50	10
	3	5	4	0			
						35	
						50	60

Our aim is to iterate toward an optimal solution, starting with this solution.

A little bit of reflection should convince you that the present scenario is essentially the same as that at the start of Phase II of the standard Simplex method. There is, however, a logistical difference, namely that the standard Simplex tableau associated with the current solution is not explicitly available to us. Therefore, we need to develop a different set of procedures for generating informations that are necessary for the execution of the Simplex algorithm. In particular, it is critical that we have corresponding mechanisms for conducting optimality tests and for performing pivots.

We begin with the question of whether or not the current solution is optimal. In the standard Simplex method, the optimality test is based on a reading of the coefficients of the nonbasic variables in the zeroth row of the Simplex tableau; that is, it is based on a reading of the reduced costs. Since the reduced costs are not explicitly available in a given transportation tableau, our first task is to develop a method for (re)constructing them.

Recall that the reduced cost associated with a nonbasic variable is defined to be the amount by which the objective-function value degrades if we increase (nominally) the value of that nonbasic variable by 1 (while holding all other nonbasic variables at 0). We will apply this definition to constructively generate the reduced costs associated with all nonbasic variables.

In a simplex tableau, if a nonbasic variable has a negative (positive) reduced cost, introducing it into the basis increases (decreases) the objective value. This is because, we let $z = \text{objective function}$ and then construct Row 0 as $z - \text{objective function} = 0$. In the transportation problem, we directly work with the objective $z = \text{objective function}$. Thus, In a transportation tableau, if a nonbasic variable has a negative (positive) reduced cost, introducing it into the basis decreases (increases) the objective value. Because of this modification, a transportation tableau is optimal if the reduced cost of each nonbasic variable is nonnegative.

Consider the nonbasic variable x_{11} , which is located in cell (1, 1). A cell that contains a nonbasic variable will be referred to as a *nonbasic cell*. Now, imagine an increase in the value of x_{11} from 0 to δ , where δ is

nonnegative. Since the sum of the x_{ij} values in row 1, i.e., the row sum $x_{11} + x_{12} + x_{13}$, must be maintained at 45 to preserve feasibility, such an increase will necessitate a corresponding decrease in x_{12} and/or x_{13} . Notice that both x_{12} and x_{13} are basic variables. A cell that contains a basic variable will be referred to as a *basic cell*. We will consider first the basic cell (1, 3). Observe that if we attempt to decrease the value of x_{13} from 30 to $30 - \delta$, then, since the column sum $x_{13} + x_{23} + x_{33}$ must be maintained at 30, at least one of the variables x_{23} and x_{33} in that column must undergo a corresponding increase. Since both x_{23} and x_{33} are nonbasic, their values are to remain at 0; it follows that a decrease in x_{13} is not permitted. This leads us back to basic cell (1, 2). Again, since the column sum $x_{12} + x_{22} + x_{32}$ must be maintained at 60, a decrease in x_{12} will necessitate a corresponding increase in x_{22} and/or x_{32} . Notice, however, that an increase in x_{32} will force a corresponding decrease in the nonbasic cells (3, 1) and/or (3, 3); therefore, an increase in x_{32} is not permitted. It follows that the only sequence of permissible revisions, at this point, is to decrease the value of x_{12} from 15 to $15 - \delta$ and then to match that decrease with an increase of the value of x_{22} from 10 to $10 + \delta$. Next, the row-sum requirement for row 2 shows that we should now decrease the value of x_{21} from 50 to $50 - \delta$. Finally, an examination of column 1 shows that we have managed to navigate through a full cycle of revisions, in the sense that this last decrease is balanced by the original increase in x_{11} .

In summary, above discussion reveals that a nominal increase in x_{11} from 0 to δ can be “accommodated” by sequentially decreasing the value of x_{12} from 15 to $15 - \delta$, increasing the value of x_{22} from 10 to $10 + \delta$, and finally decreasing the value of x_{21} from 50 to $50 - \delta$. (All other x_{ij} ’s, basic or not, retain their original values.) This sequence of revisions can be explicitly indicated as in the tableau below.

		Sinks			
		1	2	3	
	1	3	2	0	
	+ δ		15 - δ	30	45
	2	1	5	0	
Sources	50 - δ		10 + δ		60
	3	5	4	0	
			35		35
		50	60	30	

An inspection of this tableau shows that the only cells “visited” by this sequence of revisions are: (1, 1), (1, 2), (2, 2), and (2, 1). The layout of these cells can be described in the form of a path, as shown below.

$$\begin{array}{ccc}
 (1, 1)^* & \longrightarrow & (1, 2) \\
 \uparrow & & \downarrow \\
 (2, 1) & \longleftarrow & (2, 2)
 \end{array}$$

Thus, the path begins with cell (1, 1), which is marked with an asterisk to indicate that it is the entering nonbasic cell. We then visit basic cells (1, 2), (2, 2), and (2, 1) in succession; and finally, we return to cell (1, 1) from the last stop, cell (2, 1). The path just described is an example of what is called a *stepping-stone path*. This name originates from the fact that we are stepping through a sequence of basic cells, and that we can think of these basic cells as “stones” in a pond—the pond being the entire tableau. (Nonbasic cells are not considered as stones; therefore, if one (mis)steps on a nonbasic cell, then one falls into water and gets wet.)

Note that, as indicated by the arrows, the order of visits in the stepping-stone path above is clockwise. It is easily seen that reversing this direction to a counter-clockwise order will have no impact on the revised values of the x_{ij} ’s. It follows that the direction of visits is irrelevant.

A notable property of a stepping-stone path is that in the transportation tableau, it will always make a 90-degree turn after stepping on a cell. This is a consequence of the fact that we are alternatingly main-

taining the row-sum (or supply) constraints and the column-sum (or demand) constraints.

Another important observation is that exactly one stepping-stone path can originate from the nonbasic cell (1, 1). You should convince yourself about the validity of this claim by going through a careful review of the discussion above. This claim, in fact, holds in general; that is, every nonbasic cell has exactly one associated stepping-stone path.

Our discussion above can also be summarized more formally as follows. By increasing the value of δ , i.e., by bringing x_{11} into the basis, we can generate a family of solutions of the form:

$$(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}) = (\delta, 15 - \delta, 30, 50 - \delta, 10 + \delta, 0, 0, 35, 0).$$

Geometrically, this corresponds to an attempt to follow an edge of the feasible region, starting from the corner-point solution

$$(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}) = (0, 15, 30, 50, 10, 0, 0, 35, 0).$$

Since all variables are to remain nonnegative, we need to require that both $15 - \delta$ and $50 - \delta$ stay nonnegative. It follows that the value of δ should not exceed 15. With $\delta = 15$, we obtain the new solution

$$(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}) = (15, 0, 30, 35, 25, 0, 0, 35, 0);$$

and this means that we have arrived at a corner-point solution that is adjacent to the previous one. In other words, what we have done is the equivalent of an ordinary Simplex pivot.

We are now ready to derive the reduced cost associated with x_{11} . From the above tableau, we see that for any given δ , the difference between the new objective-function value and the original objective-function value can be computed by adding individual differences associated with cells on the stepping-stone path. Specifically, the individual differences are: $3 \cdot \delta$ for cell (1, 1), $2 \cdot (-\delta)$ for cell (1, 2), $5 \cdot \delta$ for cell (2, 2), and $1 \cdot (-\delta)$ for cell (2, 1). It follows that with $\delta = 1$, the overall difference can be summed up as:

$$\begin{aligned} 3 \cdot \delta + 2 \cdot (-\delta) + 5 \cdot \delta + 1 \cdot (-\delta) &= 3 \cdot 1 + 2 \cdot (-1) + 5 \cdot 1 + 1 \cdot (-1) \\ &= 3 - 2 + 5 - 1 \\ &= 5. \end{aligned}$$

Thus, a 1-unit increase in x_{11} results in a 5-unit increase (which is a degradation) in the objective-function value. In other words, the reduced cost associated with x_{11} equals 5. Since this reduced cost is positive, it is not desirable to bring x_{11} into the basis.

Suppose a variable x_{ij} is nonbasic in a solution; then, we will denote the reduced cost associated with x_{ij} by \bar{c}_{ij} . In this notation, the above calculation can be summarized more-compactly as:

$$\begin{aligned} \bar{c}_{11} &= c_{11} - c_{12} + c_{22} - c_{21} \\ &= 3 - 2 + 5 - 1 \\ &= 5, \end{aligned}$$

where the c_{ij} 's are picked up sequentially from the cells on the stepping-stone path.

There are three other nonbasic cells in the above tableau, namely (2, 3), (3, 1), and (3, 3). We now continue on to an examination of these cells. Since the underlying ideas have been explained in detail, we will be brief.

The stepping-stone path associated with cell (2, 3) is:

$$\begin{array}{ccc} (1, 2) & \longrightarrow & (1, 3) \\ \uparrow & & \downarrow \\ (2, 2) & \longleftarrow & (2, 3)^* \end{array}$$

After picking up the c_{ij} 's (and alternating their signs) in the order (2, 3), (2, 2), (1, 2), and (1, 3), the reduced cost associated with x_{23} can be computed as:

$$\begin{aligned} \bar{c}_{23} &= 0 - 5 + 2 - 0 \\ &= -3. \end{aligned}$$

The fact that this reduced cost is negative implies that the current solution is not optimal.

The stepping-stone path associated with cell (3, 1) is:

$$\begin{array}{ccc} (2, 1) & \longrightarrow & (2, 2) \\ \uparrow & & \downarrow \\ (3, 1)^* & \longleftarrow & (3, 2) \end{array}$$

By stepping through cells on this path (starting with cell (3, 1)), the reduced cost associated with x_{31} can be computed as:

$$\begin{aligned} \bar{c}_{31} &= 5 - 1 + 5 - 4 \\ &= 5. \end{aligned}$$

Since \bar{c}_{31} is positive, it is not desirable to bring x_{31} into the basis.

Finally, the stepping-stone path associated with cell (3, 3) is:

$$\begin{array}{ccc} (1, 2) & \longrightarrow & (1, 3) \\ \uparrow & & \downarrow \\ \bullet & & \bullet \\ \uparrow & & \downarrow \\ (3, 2) & \longleftarrow & (3, 3)^* \end{array}$$

Here, the symbol “•” (an icon of a crossed-out cell) between cells (3, 2) and (1, 2) denotes the fact that we do not intend to make any revision in cell (2, 2); the interpretation of the • between cells (1, 3) and (3, 3) is similar. In the language of the pond analogy, this means that we are “jumping” over cells (2, 2) and (2, 3). (In this connection, we note that, in general, the shape of a stepping-stone path can be very complex. In particular, it does not have to be rectangular.) By stepping through cells on this path, we obtain:

$$\begin{aligned} \bar{c}_{33} &= 0 - 4 + 2 - 0 \\ &= -2. \end{aligned}$$

The fact that \bar{c}_{33} is negative indicates, again, that the current solution is not optimal.

Since the value of \bar{c}_{23} is more negative than that of \bar{c}_{33} , we should now bring x_{23} into the basis. In the language of the standard Simplex algorithm, this means that we should execute a pivot in the x_{23} -column. However, since we are working with a transportation tableau, this pivot will have to be implemented in a

different manner. Observe that bringing x_{23} into the basis means that we are interested in solutions of the form indicated by the tableau below.

		Sinks			
		1	2	3	
Sources	1	3	2	0	45
			$15 + \delta$	$30 - \delta$	
	2	1	5	0	60
		50	$10 - \delta$	$+\delta$	
3	5	4	0		35
			35		
		50	60	30	

An inspection of this tableau shows that the x_{ij} values contained in cells (2, 2) and (1, 3), namely $10 - \delta$ and $30 - \delta$ are being decreased. Since all variables must remain nonnegative, it follows that the maximum possible value for δ is 10 (this corresponds to a ratio test in the ordinary Simplex algorithm); and that x_{22} is the leaving variable. We will, therefore, let $\delta = 10$ in the above tableau; and doing so takes us to the tableau below.

		Sinks			
		1	2	3	
Sources	1	3	2	0	45
			25	20	
	2	1	5	0	60
		50		10	
3	5	4	0		35
			35		
		50	60	30	

Note that cell (2, 2) is now left blank, indicating that x_{22} is nonbasic in the new solution.

The next task, of course, is to test the new solution for optimality. The nonbasic cells in the new tableau are: (1, 1), (2, 2), (3, 1), and (3, 3). The stepping-stone paths associated with these cells are:

$$\begin{array}{ccccc}
 (1, 1)^* & \longrightarrow & \bullet & \longrightarrow & (1, 3) \\
 \uparrow & & & & \downarrow \\
 (2, 1) & \longleftarrow & \bullet & \longleftarrow & (2, 3)
 \end{array}$$

for cell (1, 1);

$$\begin{array}{ccc}
 (1, 2) & \longrightarrow & (1, 3) \\
 \uparrow & & \downarrow \\
 (2, 2)^* & \longleftarrow & (2, 3)
 \end{array}$$

for cell (2, 2);

$$\begin{array}{ccccc}
 & & (1, 2) & \longleftarrow & (1, 3) \\
 & & \downarrow & & \uparrow \\
 (2, 1) & \longrightarrow & \bullet & \longrightarrow & (2, 3) \\
 \uparrow & & \downarrow & & \\
 (3, 1)^* & \longleftarrow & (3, 2) & &
 \end{array}$$

for cell (3, 1); and finally,

$$\begin{array}{ccc}
 (1, 2) & \longrightarrow & (1, 3) \\
 \uparrow & & \downarrow \\
 \bullet & & \bullet \\
 \uparrow & & \downarrow \\
 (3, 2) & \longleftarrow & (3, 3)^*
 \end{array}$$

for cell (3, 3). Therefore, the corresponding (new) reduced costs are:

$$\begin{aligned}\bar{c}_{11} &= 3 - 0 + 0 - 1 \\ &= 2,\end{aligned}$$

$$\begin{aligned}\bar{c}_{22} &= 5 - 2 + 0 - 0 \\ &= 3,\end{aligned}$$

$$\begin{aligned}\bar{c}_{31} &= 5 - 1 + 0 - 0 + 2 - 4 \\ &= 2,\end{aligned}$$

and

$$\begin{aligned}\bar{c}_{33} &= 0 - 4 + 2 - 0 \\ &= -2.\end{aligned}$$

Since \bar{c}_{33} remains negative, the current solution is not optimal. Therefore, we will next let x_{33} enter the basis.

Bringing x_{33} into the basis means that we will consider solutions of the form below.

		Sinks			
		1	2	3	
Sources	1	3	2	0	45
			$25 + \delta$	$20 - \delta$	
	2	1	5	0	
	50		10		
3	5	4	0	35	
		$35 - \delta$	$+\delta$		
		50	60	30	

An inspection of cells (1, 3) and (3, 2) shows that x_{33} can be boosted up to 20; and that at this level, x_{13} exits the basis. Execution of this pivot now yields the new solution below.

		Sinks			
		1	2	3	
Sources	1	3	2	0	45
			45		
	2	1	5	0	
	50		10		
3	5	4	0	35	
		15	20		
		50	60	30	

After constructing a new set of stepping-stone paths for cells (1, 1), (1, 3), (2, 2), and (3, 1), details of which we omit, the new reduced costs are:

$$\begin{aligned}\bar{c}_{11} &= 3 - 2 + 4 - 0 + 0 - 1 \\ &= 4,\end{aligned}$$

$$\begin{aligned}\bar{c}_{13} &= 0 - 0 + 4 - 2 \\ &= 2,\end{aligned}$$

$$\begin{aligned}\bar{c}_{22} &= 5 - 0 + 0 - 4 \\ &= 1,\end{aligned}$$

and

$$\begin{aligned}\bar{c}_{31} &= 5 - 1 + 0 - 0 \\ &= 4.\end{aligned}$$

Since all reduced costs are positive, we conclude, finally, that the current solution is optimal. The fact that these reduced costs are strictly positive also implies that there are no other optimal solutions.

In conclusion, the optimal solution produced by the above procedure, which is called the *stepping-stone method*, is:

$$(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}) = (0, 45, 0, 50, 0, 10, 0, 15, 20).$$

This basic feasible solution has an objective-function value of 200.

3.2 The $u - v$ Method

This section discusses another method to find the optimal solution. However, it is not central to further developments so it can be skipped without disturbing the continuity. Recall that in every iteration of the stepping-stone method, one has to construct a stepping-stone path for every nonbasic cell. This aspect of the method seems laborious. In fact, it can be shown that the amount of effort involved in this task grows exponentially as a function of problem size. Thus, the required computational effort for large-scaled problems will be prohibitive. It is therefore desirable to look for a more-efficient alternative. The $u-v$ method is one alternative whose computational effort grows only linearly.

Again, we will use the previous numerical example to motivate the idea behind the $u-v$ method. Recall that the solution produced by the least-cost method is given in the tableau below.

		Sinks			
		1	2	3	
Sources	1	3	2	0	45
			15	30	
	2	1	5	0	
	50	10			
3	5	4	0	35	
		35			
		50	60	30	

For reasons that will become clear shortly, let us suppose that the c_{ij} 's in this tableau are modified to a new set of values, as specified in the tableau below.

		Sinks			
		1	2	3	
Sources	1	5	0	0	45
			15	30	
	2	0	0	-3	
	50	10			
3	5	0	-2	35	
		35			
		50	60	30	

Observe that a distinct feature of this tableau is that the c_{ij} 's associated with the basic cells are all equal to 0. (That some of the costs in this tableau are negative is not a concern; this is because negative costs can be interpreted as profits.) To understand the implication of this feature, let us compute the reduced cost for cell $(1, 1)$. Recall that the stepping-stone path for cell $(1, 1)$ is:

$$\begin{array}{ccc} (1, 1)^* & \longrightarrow & (1, 2) \\ \uparrow & & \downarrow \\ (2, 1) & \longleftarrow & (2, 2) \end{array}$$

Therefore,

$$\begin{aligned}\bar{c}_{11} &= c_{11} - c_{12} + c_{22} - c_{21} \\ &= 5 - 0 + 0 - 0 \\ &= 5.\end{aligned}$$

In other words, since every basic cell along this stepping-stone path has its associated c_{ij} equal to 0, we have $\bar{c}_{11} = c_{11}$. A little bit of reflection now reveals that, in fact, we have

$$\bar{c}_{ij} = c_{ij}$$

for every nonbasic cell, regardless of the shape of the stepping-stone paths. It follows that whenever a given set of c_{ij} 's and a given basic feasible solution, together, "happen" to have the distinct feature described above, then the reduced costs can be read directly from the tableau without any need for constructing the stepping-stone paths.

While this remarkable feature is certainly desirable, it seems unrealistic to expect such "luck" in a given tableau. Surprisingly, it turns out that we actually have a lot of flexibility in specifying the c_{ij} values in a problem. What this means is that it is possible to modify a given set of c_{ij} 's in ways that preserve the identity of the optimal solution.

To see how this is accomplished, let us consider the first tableau again. Suppose all three costs in column 1 are revised downward by 1 unit. That is, let $c_{11} = 3 - 1 = 2$, $c_{21} = 1 - 1 = 0$, and $c_{31} = 5 - 1 = 4$. Clearly, as a result of this downward revision, the total contribution to the objective-function value from the three variables in the first column, namely x_{11} , x_{21} , and x_{31} , must undergo a corresponding reduction. This reduction can be explicitly calculated as:

$$\begin{aligned}(3 \cdot x_{11} + 1 \cdot x_{21} + 5 \cdot x_{31}) &- (2 \cdot x_{11} + 0 \cdot x_{21} + 4 \cdot x_{31}) \\ &= x_{11} + x_{21} + x_{31} \\ &= 50,\end{aligned}$$

where the second equality is a consequence of the demand constraint $x_{11} + x_{21} + x_{31} = 50$ at Sink 1. Notice that the outcome of this calculation is independent of the specific values of x_{11} , x_{21} , and x_{31} . In other words, *every* feasible solution will have its objective-function value reduced by 50. It follows that, indeed, if a solution is optimal prior to these cost revisions, then the same solution will remain optimal after the revisions. Similarly, as a consequence of the requirement that $x_{11} + x_{12} + x_{13} = 45$, the identity of the optimal solution is preserved after downward revisions in all three costs in row 1 by 1 unit.

Continuation of this argument shows that, in fact, we can modify the costs in every row and every column in this manner without any fear of "losing" the identity of the optimal solution. For this reason, we shall say that two sets of costs are *equivalent* if they are related to each other in this manner.

Armed with this newly-found flexibility, let us return to the "distinct feature" noted earlier. The question now is: Is it possible to modify the original c_{ij} 's to arrive at an equivalent set of costs that has this distinct feature? We will show that the answer is yes.

The idea is to work with a sequence of variably-sized reductions (as opposed to 1-unit reductions) in the costs in the transportation tableau, first row-by-row and then column-by-column. Specifically, let

u_i = the size of a reduction in every cost in row i , where $i = 1, 2, 3$.

v_j = the size of a reduction in every cost in column j , where $j = 1, 2, 3$.

We shall refer to the u_i 's and the v_j 's as the *modifiers*. Thus, u_1 is the modifier for row 1, v_1 is the modifier for column 1, and so on. We will also allow these modifiers to assume negative values. That the letters "u" and "v" are used to denote the modifiers is why we refer to this method as the *u-v method*.

Clearly, after cycling through these six modifications, the original cost in cell (i, j) will be reduced twice, the first time by u_i and the second time by v_j . It follows that the revised cost for cell (i, j) is equal to $c_{ij} - u_i - v_j$. This is

explicitly shown in the tableau below.

		Sinks				
		1	2	3	Modifier	
Sources	1	$3 - u_1 - v_1$	$2 - u_1 - v_2$	$0 - u_1 - v_3$	45	u_1
			15	30		
	2	$1 - u_2 - v_1$	$5 - u_2 - v_2$	$0 - u_2 - v_3$	60	u_2
	50		10			
	3	$5 - u_3 - v_1$	$4 - u_3 - v_2$	$0 - u_3 - v_3$	35	u_3
			35			
		50	60	30		
Modifier		v_1	v_2	v_3		

Notice that we have also indicated the modifier for every row and every column at the right and bottom margins of this tableau. It is helpful to visualize the revised cost in cell (i, j) as being equal to the original c_{ij} subtracted first by the modifier located at the right margin of row i and then by the modifier located at the bottom margin of column j .

Now, in light of the distinct feature above, our goal is to choose a set of values for the u_i 's and the v_j 's to achieve the outcome $c_{ij} - u_i - v_j = 0$, or equivalently $c_{ij} = u_i + v_j$, in every basic cell. That is, we would like to see:

$$2 = u_1 + v_2,$$

$$0 = u_1 + v_3,$$

$$1 = u_2 + v_1,$$

$$5 = u_2 + v_2,$$

and

$$4 = u_3 + v_2.$$

It follows that the desired values for the modifiers can be obtained by solving this system of 5 linear equations in 6 unknowns. Note that an important feature of this equation system is that exactly two variables appear in every equation. We now show that this feature greatly reduces the solution effort.

Since the number of variables is greater than the number of equations by 1, we have one extra "degree of freedom." This means that we can choose to assign an arbitrary value to any one of the modifiers. Let us assign, say, a 0 to u_1 . Since u_1 appears in the first two equations, which are $2 = u_1 + v_2$ and $0 = u_1 + v_3$, this initial assignment immediately implies that we have $v_2 = 2 - u_1 = 2 - 0 = 2$ and $v_3 = 0 - u_1 = 0 - 0 = 0$, respectively. Since v_2 appears in the last two equations, which are $5 = u_2 + v_2$ and $4 = u_3 + v_2$, the just-assigned value for v_2 , in turn, implies that we have $u_2 = 5 - v_2 = 5 - 2 = 3$ and $u_3 = 4 - v_2 = 4 - 2 = 2$. Finally, since u_2 appears in the only remaining equation, namely $1 = u_2 + v_1$, the just-assigned value for u_2 further implies that $v_1 = 1 - u_2 = 1 - 3 = -2$. This completes the solution process.

The set of modifiers we obtained can also be entered into the tableau as follows.

		Sinks				
		1	2	3	Modifier	
Sources	1	$3 - u_1 - v_1$	$2 - u_1 - v_2$	$0 - u_1 - v_3$	45	$u_1 = 0$
			15	30		
	2	$1 - u_2 - v_1$	$5 - u_2 - v_2$	$0 - u_2 - v_3$	60	$u_2 = 3$
	50		10			
	3	$5 - u_3 - v_1$	$4 - u_3 - v_2$	$0 - u_3 - v_3$	35	$u_3 = 2$
			35			
		50	60	30		
Modifier		$v_1 = -2$	$v_2 = 2$	$v_3 = 0$		

You should verify visually that for every basic cell (say cell (i, j)), we indeed have $c_{ij} = u_i + v_j$. That is, we have succeeded in producing a set of modifiers that transforms the original costs into an equivalent set of costs that has the

desired distinct feature.

What about the revised costs in the nonbasic cells? An inspection of the above tableau shows that the revised cost in cell (1, 1), for example, is equal to $3 - u_1 - v_1 = 3 - 0 - (-2) = 5$; and that the revised costs in other nonbasic cells can be similarly computed. The results of these calculations (including those for the basic cells) are given in the tableau below.

		Sinks			
		1	2	3	
Sources	1	5	0	0	45
	2	0	0	-3	60
	3	5	0	-2	35
		50	60	30	

Notice that this tableau is precisely the one that motivated our development of the u - v method at the beginning of this discussion.

Finally, recall that with the distinct feature in place, the reduced cost associated with a nonbasic cell is simply the cost in that cell, without any need for constructing an explicit stepping-stone path. Since the reduced costs are denoted by \bar{c}_{ij} , it follows that for every nonbasic cell (say cell (i, j)), we have

$$\bar{c}_{ij} = c_{ij} - u_i - v_j.$$

Our conclusion, therefore, is that all of the reduced costs associated with a transportation tableau can be generated by an application of the u - v method. A little bit of reflection should also convince you that this procedure is computationally more efficient than the stepping-stone method.

The rest of the solution procedure for this transportation problem proceeds in the same manner as what was done earlier using the stepping-stone method. Note, however, that after the selection of an entering cell based on a computed set of modifiers, it is still necessary to construct the stepping-stone path associated with that particular cell to conduct a pivot. For completeness, we provide a brief summary below.

An inspection of the above tableau shows that the entering cell is (2, 3). The stepping-stone path associated with this cell is:

$$\begin{array}{ccc} (1, 2) & \longrightarrow & (1, 3) \\ \uparrow & & \downarrow \\ (2, 2) & \longleftarrow & (2, 3)^* \end{array}$$

After conducting a pivot according to this path, we obtain the new solution below.

		Sinks			Modifier	
		1	2	3		
Sources	1	3	2	0	45	$u_1 = 0$
	2	1	5	0	60	$u_2 = 0$
	3	5	4	0	35	$u_3 = 2$
		50	60	30		
Modifier		$v_1 = 1$	$v_2 = 2$	$v_3 = 0$		

Notice that a new set of modifiers has been specified on the right and bottom margins of this tableau. These are computed as follows. We begin by entering the assignment $u_1 = 0$ (which is arbitrarily chosen) at the right margin of row 1. Observe that cells (1, 2) and (1, 3) in row 1 are basic. Therefore, with u_1 given as 0, we should assign $v_2 = 2 - 0 = 2$ and $v_3 = 0 - 0 = 0$. These new assignments are entered at the bottom margins of column 2 and column 3, respectively. Since cell (3, 2) in column 2 is basic, the fact that $c_{32} = 4$ and $v_2 = 2$ implies that $u_3 = 4 - 2 = 2$,

which is entered at the right margin of row 3. Similarly, since cell (2, 3) in column 3 is basic, the fact that $c_{23} = 0$ and $v_3 = 0$ implies that $u_2 = 0 - 0 = 0$, which is entered at the right margin of row 2. Finally, since cell (2, 1) in row 2 is basic, the only remaining assignment is $v_1 = 1 - 0 = 1$, entered at the bottom of column 1.

With the set of new modifiers given, the reduced costs in the nonbasic cells can now be calculated as follows:

$$\begin{aligned}\bar{c}_{11} &= c_{11} - u_1 - v_1 \\ &= 3 - 0 - 1 \\ &= 2,\end{aligned}$$

$$\begin{aligned}\bar{c}_{22} &= c_{22} - u_2 - v_2 \\ &= 5 - 0 - 2 \\ &= 3,\end{aligned}$$

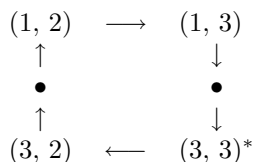
$$\begin{aligned}\bar{c}_{31} &= c_{31} - u_3 - v_1 \\ &= 5 - 2 - 1 \\ &= 2,\end{aligned}$$

and

$$\begin{aligned}\bar{c}_{33} &= c_{33} - u_3 - v_3 \\ &= 0 - 2 - 0 \\ &= -2.\end{aligned}$$

Note that these reduced costs are in agreement with those produced by the stepping-stone method earlier.

Since \bar{c}_{33} is negative, the next entering cell is (3, 3). The stepping-stone path associated with this cell is:



After conducting a pivot according to this path and updating the modifiers, we obtain the new solution below.

		Sinks						
		1	2	3				
Sources	1	3	2	0	45	$u_1 = 0$		
	2	1	5	0			60	$u_2 = 2$
	3	5	4	0				
		50	60	30				
Modifier		$v_1 = -1$	$v_2 = 2$	$v_3 = -2$				

The reduced costs associated with the nonbasic cells can be calculated as follows.

$$\begin{aligned}\bar{c}_{11} &= 3 - 0 - (-1) \\ &= 4,\end{aligned}$$

$$\begin{aligned}\bar{c}_{13} &= 0 - 0 - (-2) \\ &= 2,\end{aligned}$$

$$\begin{aligned}\bar{c}_{22} &= 5 - 2 - 2 \\ &= 1,\end{aligned}$$

and

$$\begin{aligned}\bar{c}_{31} &= 5 - 2 - (-1) \\ &= 4.\end{aligned}$$

Since all of these are positive, we conclude, finally, that the current solution is (uniquely) optimal.

4 The Streamlined Simplex Method: An Example

This section discusses an example to find the optimal solution using the Vogel's method and the $u - v$ method. However, it is not central to further developments so it can be skipped without disturbing the continuity. Consider the (minimization) transportation problem below.

		Sinks				
		1	2	3	4	
Sources	1	10	0	20	11	15
	2	12	7	9	20	25
	3	0	14	16	18	5
		5	15	15	10	

We will solve this problem using the streamlined Simplex algorithm for transportation problems. In the first phase, we will apply the Vogel's method to construct an initial basic feasible solution; and in the second phase, where the task is to iterate toward an optimal solution, we will apply the $u - v$ method to conduct optimality tests. Since the ideas behind these methods have already been explained in detail, our intent is to use this example to illustrate the complete algorithm. We will, therefore, be brief.

The first observation is that the total supply and the total demand are both equal to 45. Therefore, we have a balanced transportation problem, and there is no need to add either a dummy source or a dummy sink. In addition, note that there are 3 sources and 4 sinks; therefore, the number of basic variables in every basic feasible solution is 6 ($= 3 + 4 - 1$).

We now begin the construction of an initial basic feasible solution, using the Vogel's method. An inspection of the c_{ij} 's in the tableau above shows that the row penalties are: 10, 2, and 14; and that the column penalties are: 10, 7, 7, and 7. These are shown on the margins of the tableau below.

		Sinks					Penalty
		1	2	3	4		
Sources	1	10	0	20	11	15	10
	2	12	7	9	20	25	2
	3	0	14	16	18	5	14
		5	15	15	10		
Penalty		10	7	7	7		

Since the highest penalty comes from row 3, the first entering cell is cell (3, 1). After assigning 5 as x_{31} and going

through a round of updates, we obtain the new tableau below.

		Sinks					
		1	2	3	4		
Sources	1	10	0	20	11	15	10
	2	12	7	9	20		
	3*	0	14	16	18	5 → 0	14
		5					
		5 → 0	15	15	10		
Penalty		10 → 2	7	7 → 11	7 → 9		

Note that we have removed row 3, while leaving column 1 available for further assignment. Moreover, we have also updated the penalties.

Observe that column 3 now has the highest penalty; therefore, the next entering cell is cell (2, 3). After assigning 15 as x_{23} and updating in the same manner, we obtain the tableau below.

		Sinks					
		1	2	3*	4		
Sources	1	10	0	20	11	15	10
	2	12	7	9	20		
	3*	0	14	16	18	5 → 0	14
		5					
		5 → 0	15	15 → 0	10		
Penalty		10 → 2	7	7 → 11	7 → 9		

Since row 1 has the highest penalty, the next entering cell is cell (1, 2). After assigning 15 as x_{12} and updating, we obtain the tableau below.

		Sinks					
		1	2	3*	4		
Sources	1*	10	0	20	11	15 → 0	10
	2	12	7	9	20		
	3*	0	14	16	18	5 → 0	14
		5					
		5 → 0	15 → 0	15 → 0	10		
Penalty		10 → 2	7	7 → 11	7 → 9		

With only three cells in row 2 remaining, we simply assign, sequentially, 0 as x_{22} , 0 as x_{21} , and 10 as x_{24} . This yields the initial basic feasible solution given in the tableau below.

		Sinks					
		1*	2*	3*	4*		
Sources	1*	10	0	20	11	15 → 0	10
	2	12	7	9	20		
	3*	0	14	16	18	5 → 0	14
		5					
		5 → 0 → 0	15 → 0 → 0	15 → 0	10 → 0		
Penalty		10 → 2	7	7 → 11	7 → 9		

Note that we have entered two degenerate basic variables, in cells (2, 1) and (2, 2); and that, after entering $x_{24} = 10$, we chose to remove column 4, as opposed to removing row 2. At this point, we have completed the first phase of the algorithm. It is prudent to confirm that we indeed have 6 basic variables.

We now begin the second phase of the algorithm. The first step is to calculate the set of modifiers associated with the above initial basic feasible solution. The results are shown in the tableau below.

		Sinks					
		1	2	3	4		
		10	0	20	11		
1			15			15	$u_1 = -7$
		12	7	9	20		
2	0		0	15	10	25	$u_2 = 0$
		0	14	16	18		
3	5					5	$u_3 = -12$
		5	15	15	10		
Modifier		$v_1 = 12$	$v_2 = 7$	$v_3 = 9$	$v_4 = 20$		

The detailed calculations are as follows. We begin by entering the assignment $u_2 = 0$. This choice is motivated by the fact that all four cells in row 2 are basic. Indeed, with $u_2 = 0$, we immediately have $v_1 = 12$, $v_2 = 7$, $v_3 = 9$, and $v_4 = 20$ (which are simply copies of c_{21} , c_{22} , c_{23} , and c_{24}). Since cell (3, 1) in column 1 is basic, the fact that $c_{31} = 0$ and $v_1 = 12$ implies that $u_3 = -12$. Finally, since cell (1, 2) in column 2 is basic, our last assignment is $u_1 = 0 - 7 = -7$.

With the modifiers given, the reduced costs in the nonbasic cells can now be calculated as follows:

$$\begin{aligned} \bar{c}_{11} &= 10 - (-7) - 12 \\ &= 5, \end{aligned}$$

$$\begin{aligned} \bar{c}_{13} &= 20 - (-7) - 9 \\ &= 18, \end{aligned}$$

$$\begin{aligned} \bar{c}_{14} &= 11 - (-7) - 20 \\ &= -2, \end{aligned}$$

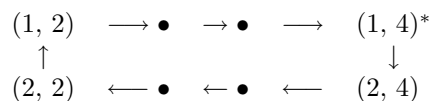
$$\begin{aligned} \bar{c}_{32} &= 14 - (-12) - 7 \\ &= 19, \end{aligned}$$

$$\begin{aligned} \bar{c}_{33} &= 16 - (-12) - 9 \\ &= 19, \end{aligned}$$

and

$$\begin{aligned} \bar{c}_{34} &= 18 - (-12) - 20 \\ &= 10. \end{aligned}$$

Since \bar{c}_{14} is negative, the next entering cell is cell (1, 4). The stepping-stone path associated with this cell is:



After conducting a pivot according to this path and updating the modifiers, we obtain the new solution below.

		Sinks					
		1	2	3	4		
Sources	1	10	0	20	11	15	$u_1 = -7$
	2	12	7	9	20	25	$u_2 = 0$
	3	0	14	16	18	5	$u_3 = -12$
		5	15	15	10		
Modifier		$v_1 = 12$	$v_2 = 7$	$v_3 = 9$	$v_4 = 18$		

It is easily seen that the reduced costs associated with the nonbasic cells are: $\bar{c}_{11} = 5$, $\bar{c}_{13} = 18$, $\bar{c}_{24} = 2$, $\bar{c}_{32} = 19$, $\bar{c}_{33} = 19$, and $\bar{c}_{34} = 12$. Since all of these are positive, we conclude, finally, that the current solution is (uniquely) optimal. The objective-function value of this solution is 315. This completes the second phase of the algorithm.

5 The Assignment Problem

Suppose that a company has m warehouses and m retailers. We want to assign warehouses to retailers. Exactly one warehouse can be assigned to a given retailer and similarly exactly one retailer can be assigned to a given warehouse. The material flow is only allowed between warehouses and retailers that are assigned to each other. We assume that the cost of sending materials from warehouse i to retailer j is c_{ij} , irrespective of how much materials are sent. In other words, we assume that the amount of flow between any warehouse and retailer pair is constant and already captured in c_{ij} .

We are interested in determining the lowest cost assignment of retailers to warehouses keeping in mind that every retailer (warehouse) has to be assigned to a warehouse (retailer). We will name this problem as *Assignment problem*.

After some reflection, we see that the assignment problem is closely related to the transportation problem. Actually every assignment problem can be thought as a transportation problem where both supplies and demands are 1 unit. With this observation, the LP formulation for the assignment problem is easily deduced from the formulation of the transportation problem as:

$$\begin{aligned}
 &\text{Minimize} && \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \\
 &\text{Subject to:} && \\
 &&& \sum_{j=1}^m x_{ij} = 1 \quad \text{for } i = 1, 2, \dots, m \quad (1) \\
 &&& \sum_{i=1}^m x_{ij} = 1 \quad \text{for } j = 1, 2, \dots, m \quad (2) \\
 &&& x_{ij} \geq 0 \quad \text{for } i = 1, 2, \dots, m \text{ and } j = 1, 2, \dots, m.
 \end{aligned}$$

Equality constraint (1) says that warehouse i can be assigned exactly once and similarly (2) says that retailer j can be assigned exactly once.

For a 3 warehouse and 3 retailer example, a feasible solution

$$(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}) = (0, 1, 0, 1, 0, 0, 0, 0, 1)$$

implies that warehouse 1, warehouse 2 and warehouse 3 are assigned to retailer 2, retailer 1 and retailer 3 respectively. The cost of such an assignment would be $c_{12} + c_{21} + c_{33}$.

How about a solution like

$$(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}) = (0, 1/2, 1/2, 1/2, 1/2, 0, 1/2, 0, 1/2)$$

Since this solution has fractional numbers it is not clear what the assignment is. Indeed, this solution does not give us a feasible assignment. Then we have to avoid such fractional solutions while solving the assignment problem. One way to do this is to declare all variables to be integers, however then we would be in the integer programming domain.

It turns out that for the assignment problem, all basic feasible solutions will be either 0 or 1. The argument for this fact is beyond the scope of this course. However, one can take a small assignment problem say 3×3 and perform couple simplex iterations to see this. Especially, the fact that the pivot row never needs to be divided by an integer is striking. In summary, we do not have to set variables to be integer. Variables will be integer because of the special structure of the constraints.

We will now give a 3×3 instance of assignment problem. To define this specific instance, we let $m = 3$ and

$$\mathbf{c} = \begin{bmatrix} 10 & 12 & 8 \\ 6 & 8 & 10 \\ 6 & 4 & 8 \end{bmatrix}.$$

That is assigning warehouse 1 to retailer 1 costs 10, to retailer 2 costs 12 and to retailer 3 costs 8, etc. Then the LP formulation becomes:

$$\begin{array}{llllllllll} \text{Minimize} & 10x_{11} & +12x_{12} & +8x_{13} & +6x_{21} & +8x_{22} & +10x_{23} & +6x_{31} & +4x_{32} & 8x_{33} \\ \text{Subject to:} & & & & & & & & & \\ & x_{11} & +x_{12} & +x_{13} & & & & & & & = 1 \\ & & & & x_{21} & +x_{22} & +x_{23} & & & & = 1 \\ & & & & & & & x_{31} & +x_{32} & +x_{33} & = 1 \\ & x_{11} & & & +x_{21} & & & +x_{31} & & & = 1 \\ & & x_{12} & & & +x_{22} & & & +x_{32} & & = 1 \\ & & & x_{13} & & & +x_{23} & & & +x_{33} & = 1 \\ & & & & & & & x_{ij} \geq 0 & \text{for } i = 1, 2, 3 \text{ and } j = 1, 2, 3 \end{array}$$

As we pointed out earlier, this formulation can be solved using Simplex or the stepping-stone method. There are also special solution techniques designed for the assignment algorithm but we will not study those for the interest of the time.

6 The Shortest Path Problem

Suppose that somebody plans to drive from Ithaca, NY to Dallas, TX (this is not as uncommon as you may think, see the movie “Road Trip”). There are several possible routes. One possible route is Ithaca \rightarrow Cleveland \rightarrow Columbus \rightarrow Cincinnati \rightarrow Louisville \rightarrow Nashville \rightarrow Memphis \rightarrow Little Rock \rightarrow Dallas. In an alternative route, one may go through Columbus \rightarrow Indianapolis \rightarrow Louisville instead of Columbus \rightarrow Cincinnati \rightarrow Louisville.

It is often the case that one wants to take the shortest route for such a trip. The length of a route is the distance traveled between Ithaca and Dallas while traversing that route. Denoting the cities with their first two letters It \rightarrow Cl \rightarrow Co \rightarrow Ci \rightarrow Lo \rightarrow Na \rightarrow Me \rightarrow Li \rightarrow Da route has the length of about 2500 km.

This length is found by summing up the distances between the cities visited on this route. The problem of finding the shortest route between It and Da is a shortest path problem.

We will now define the shortest path problem formally. Suppose that we are given a network with a set of nodes and arcs where an arc going from node i to node j has a length denoted by c_{ij} (≥ 0). A path is defined as a sequence of arcs. The length of a path is the sum of the lengths of the arcs in the path. Clearly, there could be many paths between any given two nodes. The shortest path problem deals with finding the path with the shortest length.

Going back to driving from It to Da, cities are nodes, highways are arcs, length of each highway is the length of the corresponding arc. Each route from It to Da is a path. We will next find the shortest path from It to Da.

In order to find the shortest path from It to Da, it seems we need to find all the shortest paths from It to other cities. Now suppose that we want to find all such paths and their lengths. For the remaining discussion we number the nodes from 1 to 8, where 1 is Ithaca (the source node) and 8 is Dallas (the destination node). Let u_j be the length of the shortest path from node 1 (Ithaca) to node j (City j). Naturally, $u_1 = 0$.

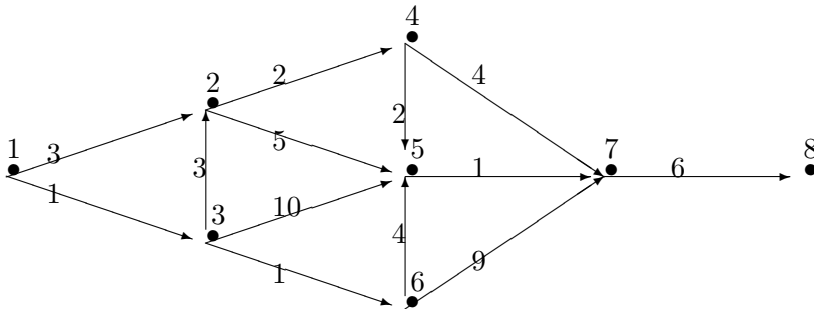
After some thinking we observe that every piece of a shortest path is a shortest path itself; Consider a shortest path \mathcal{P} from 1 to j that goes through k . 1- k portion of path \mathcal{P} is a shortest path from 1 to k . Also k - j portion of path \mathcal{P} is another shortest path from k to j . In other words, if either 1- k or k - j portion of \mathcal{P} is not a shortest path, then \mathcal{P} itself can not be a shortest path. This observation is known as the “principle of optimality”.

For each node j there must be some final arc (k, j) in a shortest path from 1 to j . Then, without knowing the identity of k , we can write that $u_j = u_k + c_{kj}$. This equality follows from the fact that the portion of the path P which extends only up to k is a shortest path, i.e., the principle of optimality. Since u_j is the length of the shortest path, we want to choose k so that $u_j = u_k + c_{kj}$ is as small as possible. Thus,

$$u_j = \min_{k \neq j} \{u_k + c_{kj}\} \quad (1).$$

We will now provide an algorithm to compute the shortest path. During the algorithm we will keep a list of nodes R . For every node j in R , we will know u_j for sure. If a node is not in R , we set $u_j = \infty$. Initially $R = \{1\}$, $u_1 = 0$ and $u_j = \infty$ for $j \neq 1$. At every iteration we compute new u_j values for each node outside R using equation (1). Then we find, the node (outside R) with the smallest u_j value and add it to the list R . At every iteration we add exactly one node to R . As soon as the destination node is in R , we can stop and take u value of the destination node as the length of the shortest path from source to the destination.

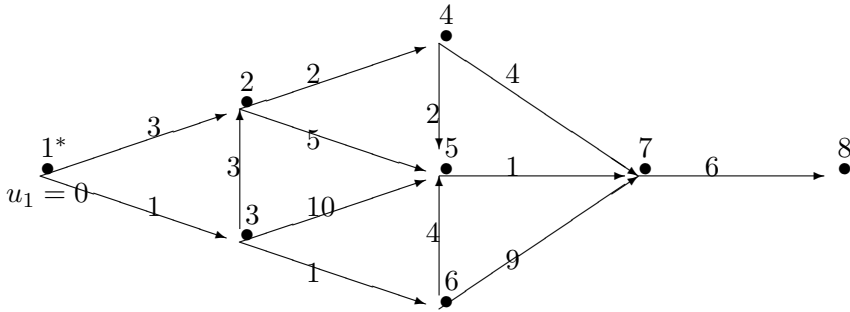
As a numerical example, we will find the shortest path from node 1 to node 8 in the following network. In this network 1=Ithaca, 2=Columbus, 3=Pittsburgh, 4=Evansville, 5=Nashville, 6=Lexington, 7=Memphis and 8=Dallas. Numbers on the arcs are arc lengths.



Initially $R = \{1\}$, $u_1 = 0$ and $u_2 = u_3 = u_4 = u_5 = u_6 = u_7 = u_8 = \infty$. Since $u_k = \infty$ if $k \notin R$, we do not need to consider $k \notin R$ in our calculations:

$$u_j = \min_{k \in R} \{u_k + c_{kj}\} \quad j \notin R.$$

We will put * on a node when it enters the set R and specify its distance from node 1.

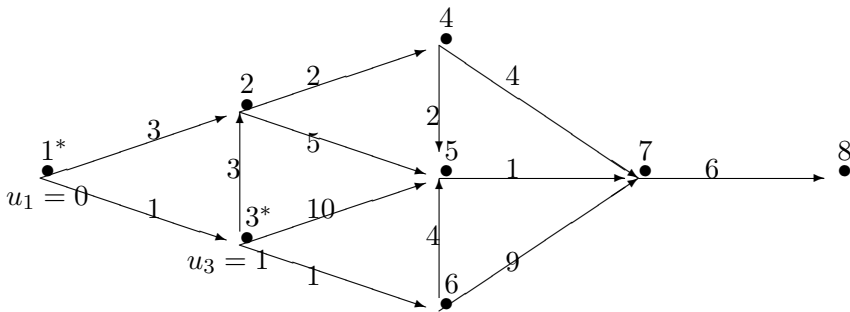


Now compute distances:

$$u_2 = \min\{u_1 + c_{12}\} = \min\{0 + 3\} = 3$$

$$u_3 = \min\{u_1 + c_{13}\} = \min\{0 + 1\} = 1$$

Observe that the rest of the nodes can not be reached via a single arc from the nodes in R , so $u_4 = u_5 = u_6 = u_7 = u_8 = \infty$. Node 3 has the smallest u_j value, it is 1. Then we add node 3 to the list R and the first iteration is completed. $R = \{1, 3\}$, $u_1 = 0$, $u_3 = 1$. In this iteration we specify only u_3 , the shortest distance to 3. Shortest distances to nodes outside R are all infinite so they are not shown in the network below:



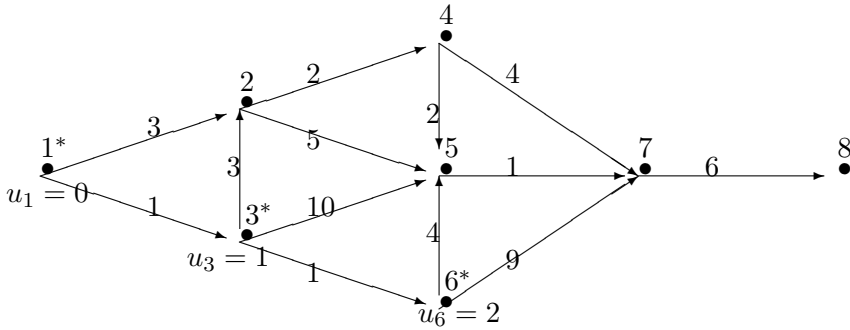
At the beginning of the second iteration, we compute the distances once more:

$$u_2 = \min\{u_1 + c_{12}, u_3 + c_{32}\} = \min\{0 + 3, 1 + 3\} = 3$$

$$u_5 = \min\{u_3 + c_{35}\} = \min\{1 + 10\} = 11$$

$$u_6 = \min\{u_3 + c_{36}\} = \min\{1 + 1\} = 2$$

Since we cannot reach nodes 4 and 7 with a single arc from R , $u_4 = u_7 = u_8 = \infty$. $u_6 = 2$ is the minimum u value so we add node 6 to R . We complete the second iteration by specifying the value of u_6 . Now, $R = \{1, 3, 6\}$, $u_1 = 0$, $u_3 = 1$, $u_6 = 2$.



In the third iteration, we compute the distances again:

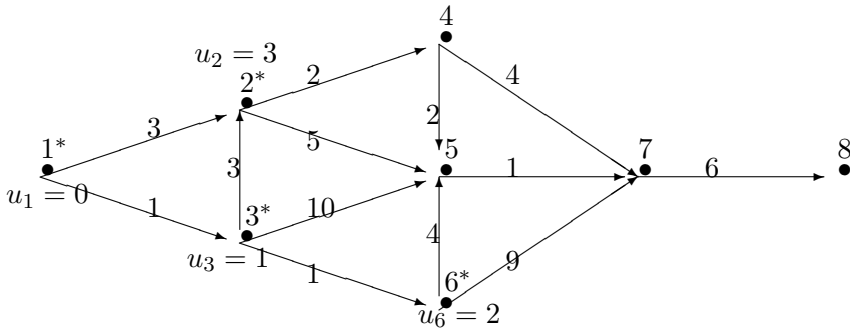
$$u_2 = \min\{u_1 + c_{12}, u_3 + c_{32}\} = \min\{0 + 3, 1 + 3\} = 3$$

$$u_5 = \min\{u_3 + c_{35}, u_6 + c_{65}\} = \min\{1 + 10, 2 + 4\} = 6$$

$$u_7 = \min\{u_6 + c_{67}\} = \min\{2 + 9\} = 11$$

$$u_4 = u_8 = \infty$$

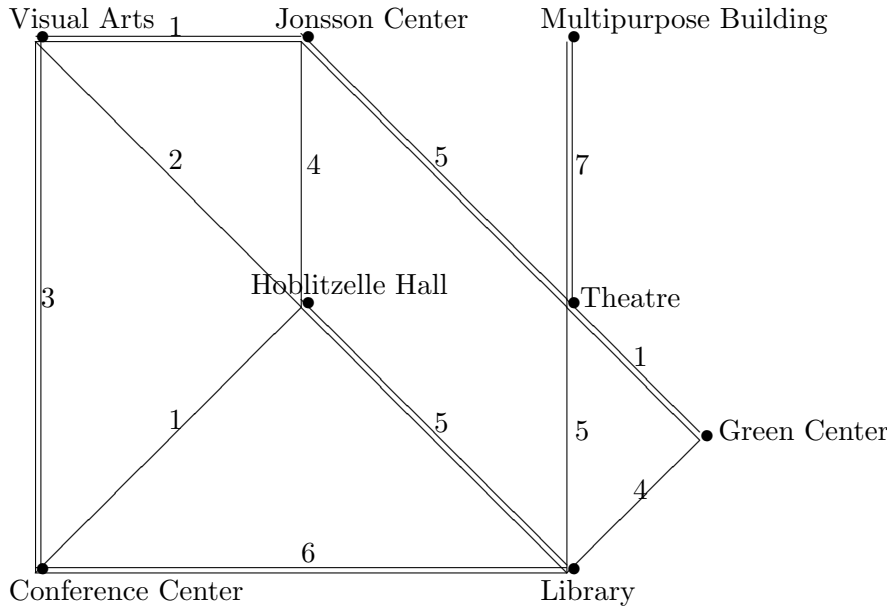
Clearly node 2 needs to be added to R . $R = \{1, 2, 3, 6\}$, $u_1 = 0$, $u_2 = 3$, $u_3 = 1$, $u_6 = 2$.



We are not going to finish up this procedure. Note that the u value of the last node added to R is always larger than or equal to the u values of the nodes already in R . Think of R as a balloon centered at node 1. At every iteration we are blowing air into the balloon so it becomes larger. At the end of first iteration its radius is 1 ($=u_3$) so it touches node 3. At the end of the second iteration, its radius is 2 ($=u_6$) so it touches node 6, and so on. When the balloon touches the node 8, we stop and take the radius of the balloon as the shortest distance from node 1 to node 8.

7 The Minimum Spanning Tree Problem

Suppose that UTD's internet / e-mail connection with the rest of the world goes through a main server at the Jonsson building. The computers in the rest of the campus must be connected to this server via communication links so that they can send / receive data to / from outside the campus. For convenience, we will not deal with the communication links inside buildings. Namely, we assume that all computers inside a building is connected to the main server if that building is connected to the Jonsson center. Below is a map of UTD campus.



On this map double lines indicate installed communication lines whereas single lines are potential links that are not installed. Since the data flow can be into or out of the main server, the arcs do not have directions; they work both ways. In this installment scheme, the computers at Hoblitzelle Hall connect to the main server via Library, Conference Center and Visual Arts building because, the link between Hoblitzelle hall and the Jonsson Center is not installed. The numbers next to each arc indicate the cost in (\$100 K) of installing the corresponding communication link. The current installment costs $1+3+6+5+5+1+7=\$28$ ($\times 10$ K), or \$280,000. We will obtain a smaller cost installation by posing this problem as a minimum spanning tree problem.

Let us first define the concept of a “tree” in a network. A tree is a set of $n - 1$ arcs connecting all the nodes in a network of n nodes. In our map, we have 8 nodes, so every tree must have exactly 7 arcs and connect all 8 nodes. Then, the set of double lines is a tree. A spanning tree touches all the nodes of a network. For us all trees are spanning trees. In this context, the cost of a tree is the sum of the costs of the arcs in the tree. We say that the cost of the tree (of double lines) in the above map is \$280,000. A tree is called *minimum spanning tree* if the cost of the tree is smaller than or equal to the cost of any other tree.

We will now introduce an algorithm to solve the minimum spanning tree problem. At each iteration of the algorithm, we keep a list of the arcs T and add arcs to this list. While adding arcs to T we pay attention to two criteria. We always choose the arc with minimum cost provided that it will connect the nodes, which are not connected currently. That is, we do not add an arc if it is a link between the nodes already linked with the arcs in T . Clearly, linking already linked nodes can only increase the costs so such a solution cannot be optimal. We stop this process once all the nodes are linked.

We will now solve the computer connection problem at the UTD campus. Initially, we suppose that there are no links installed (no double line arcs) so $T = \emptyset$:

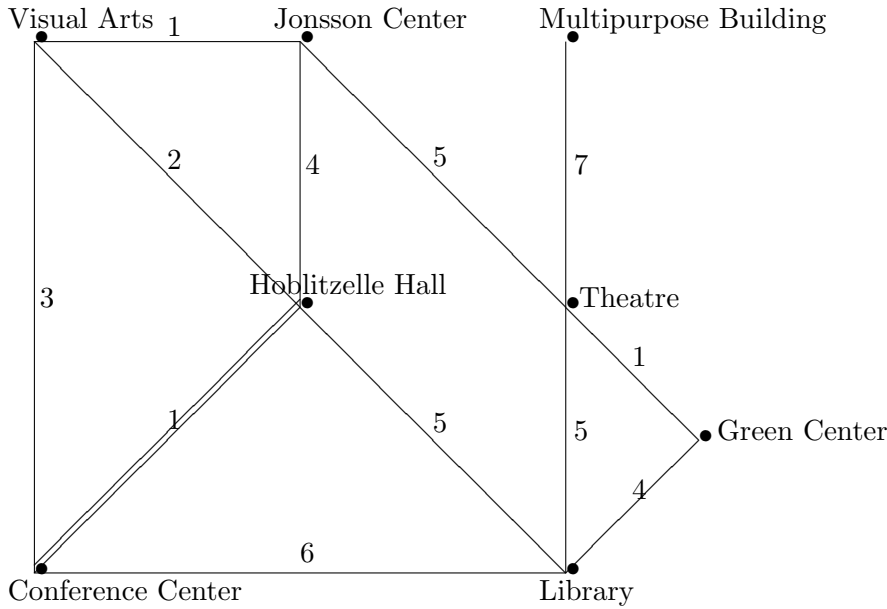
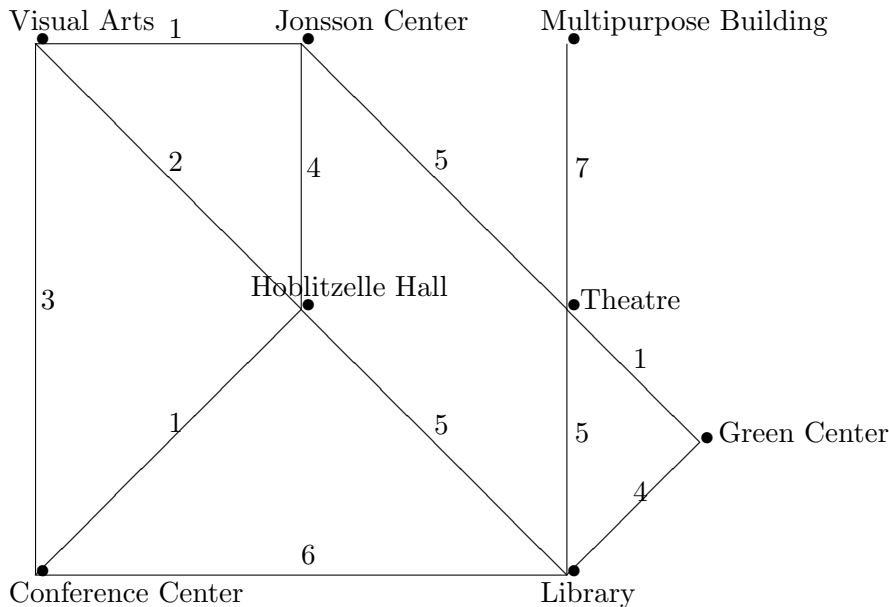


Figure 2: $T = \{H - C\}$ after the first iteration



We search for the minimum cost arc. Actually there is a tie; H-C (Hoblitzelle Hall-Conference Center), T-G (Theatre-Green Center) and V-J (Visual Arts-Jonsson center) all have cost of 1. We break ties arbitrarily and pick H-C arc to add to T . After this addition the first iteration is completed with $T = \{H - C\}$ and the network is shown in Figure 2.

In the second iteration, we again search for the minimum cost arc. Clearly, there is still a tie between T-G and V-J arcs. We arbitrarily pick T-G arc. T-G is not connecting already connected C and H so it can be added to T . Then $T = \{H - C, T - G\}$ and the second iteration is complete. In the third iteration we once more search the minimum cost arc, it is V-J. Moreover, V-J is not connecting any of the already connected nodes, so it can be added to T . At the end of the third iteration, $T = \{H - C, T - G, V - J\}$

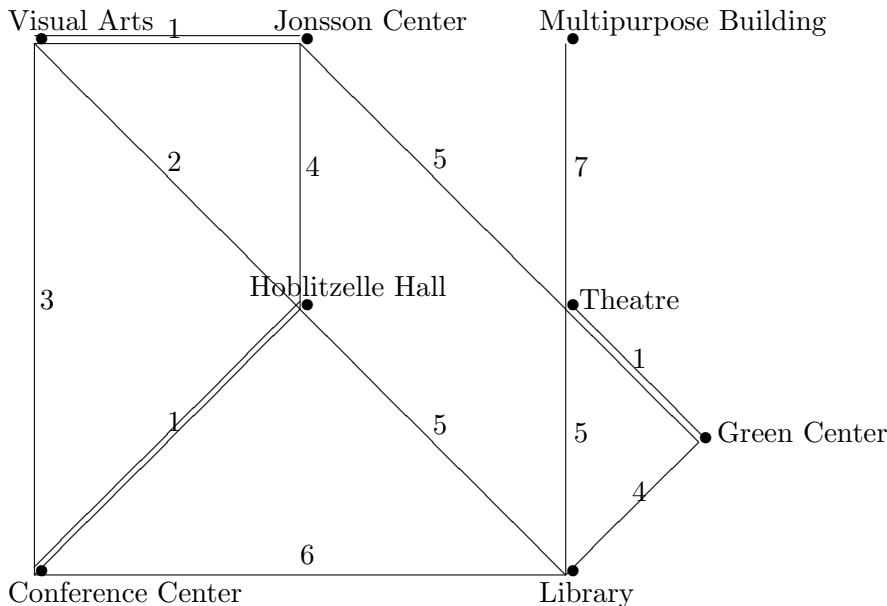


Figure 3: $T = \{H - C, T - G, V - J\}$ after the third iteration

and the network is shown in Figure 3.

In iteration four, arc $V - G$ has the minimum cost and does not connect already connected arcs. Then $T = \{H - C, T - G, V - J, V - G\}$ and the network is shown in Figure 4.

At the beginning of the fifth iteration we search for the minimum cost arc, it is $V - C$. $V - C$ cannot be added to T because it is connecting already connected Visual Arts and Conference Center (they are connected via Hoblitzelle Hall, see Figure 4.) Thus, we discard $V - C$ and look for the next minimum cost arc. Both $J - H$ and $L - G$ are eligible with arc costs 4. We arbitrarily choose to consider $J - H$ first however, it is connecting already connected J and H (see Figure 4). Thus, we discard $J - H$ as well. On the other hand $L - G$ can be added to T . At the end of the fifth iteration, $T = \{H - C, T - G, V - J, V - H, L - G\}$ and the network is shown in Figure 5.

The steps of the minimum spanning tree algorithm are straightforward so we will defer the completion of this example to Exercises.

If we look at the set of arcs in T at the end of each iteration, we see a picture similar to the progression of cracks on ice. Suppose that the nodes in our network are points on a frozen lake. The arcs are thin ice lines that connect those points. The strength of each ice line is proportional to the corresponding arc cost. Suppose that when the ice cracks, it cracks at these thin lines and the weakest line cracks first. According to this analogy, $H - C$ line cracks in the first iteration, $T - G$ line cracks in the second iteration, $V - J$ line cracks in the third iteration, $V - H$ line cracks in the fourth iteration. Note that $L - G$ line cracks in the fifth iteration although it is stronger than the $V - C$ line. To make the analogy fully correct, we have to qualify it further; The progression of T looks like cracking but never breaking ice. Note that if $V - C$ cracks in the fifth iteration, $V - C - H$ triangle will break free. Thus saying that the ice does not break free is equivalent to ruling out arcs connecting already connected nodes — while entering arcs into T .

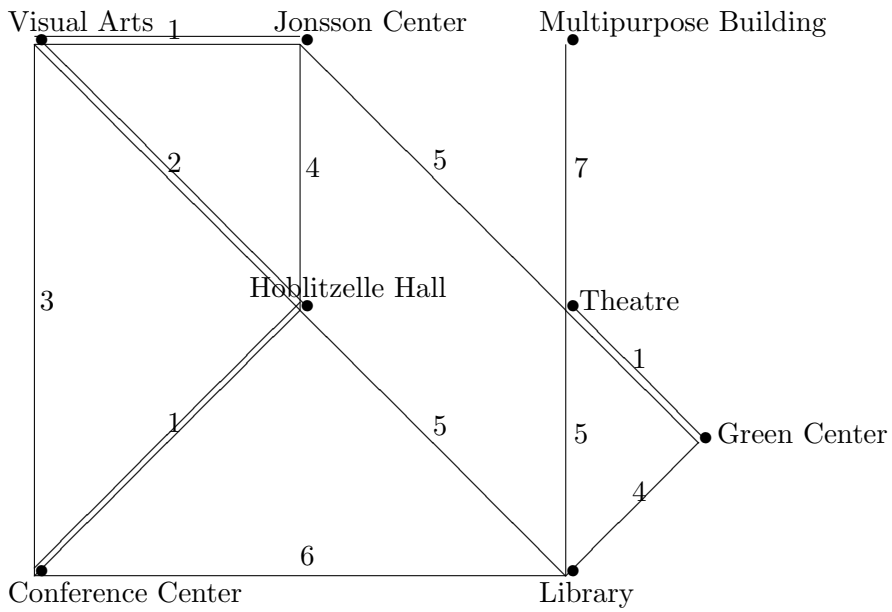


Figure 4: $T = \{H - C, T - G, V - J, V - H\}$ after the fourth iteration

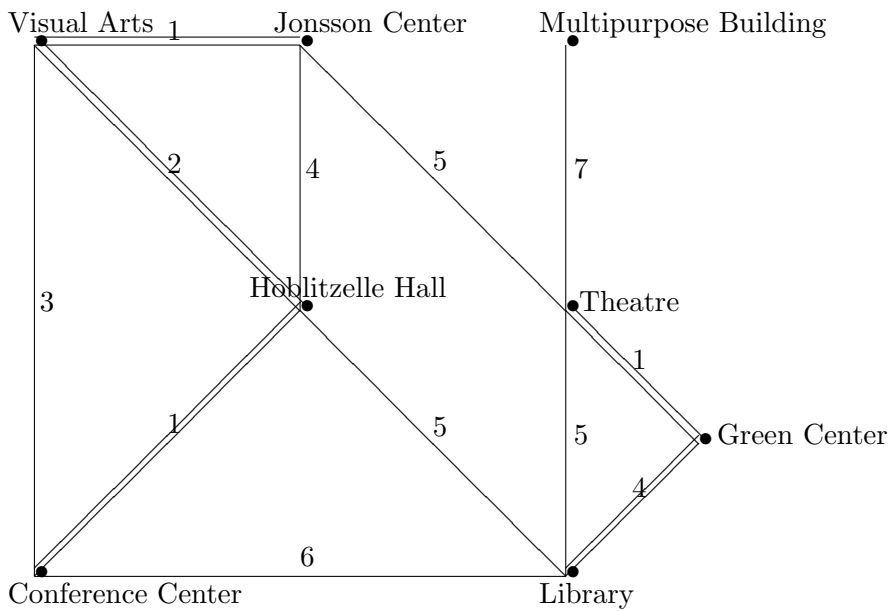


Figure 5: $T = \{H - C, T - G, V - J, V - H, L - G\}$ after the fifth iteration

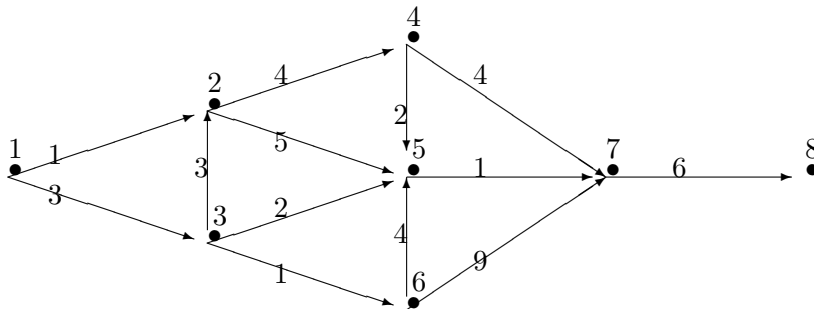
8 Exercises

1. Finish up the shortest path example and find the shortest path from Ithaca to Dallas. Illustrate the computation of u values and the nodes added to R at each iteration.
2. Finish up the minimum spanning tree example. Indicate T at each iteration and find the cost of optimal computer connection.
3. Three men (1,2,3) and three women (A,B,C) go to a ball together. Each man and woman has a preference regarding with whom they want to dance all night. The preferences (higher number indicates a stronger preference) are given in the table below:

	A	B	C
1	6	4	2
2	5	1	3
3	2	5	6

For example, 1 prefers to dance with A to dance with B or C (preferences are 6 vs. 4 or 2). B prefers to dance with 3 to dance with 1 or 2 (preferences are 5 vs. 4 or 1). Formulate a linear program to maximize the sum of the preferences of couples.

4. Find an initial basic feasible solution using the least cost method for the transportation problem in Textbook H-L: 8.2-8 on p. 396.
5. Again refer to the transportation problem in Textbook H-L: 8.2-8 on p. 396, make up the transportation tableau and write the following solution into the tableau: $x_{11} = 3$, $x_{12} = 2$, $x_{22} = 0$, $x_{32} = 1$, $x_{23} = 2$ and $x_{34} = 2$.
 - a) Explain why this solution is a basic feasible solution.
 - b) List the nonbasic variables.
 - c) Compute the reduced costs for each nonbasic variable.
 - d) Is this solution optimal, why?
 - e) Do a single simplex iteration on the transportation tableau if you say “no” to d).
6. Find the shortest path to node 8. Clearly indicate u values in every iteration.



7. TaxWiz has three auditors. Fiona and Sridhar can work at most 60 hours per week, and Josh works at most 50 hours per week. 3 projects must be completed in the next week. Each project requires several hours of work as given in the table below. The table also contains how much clients can be billed per hour by assigning an auditor to a project.

Project	I	II	III
Hours	90	30	30
Fiona	80	90	110
Sridhar	120	100	90
Josh	100	120	110

Formulate a transportation problem to maximize next week's revenue for TaxWiz.

8. Solve the previous problem to optimality using Excel or Lindo.
9. Consider the following transportation tableau:

	<i>A</i>	<i>B</i>	<i>C</i>	
1	4	2	2	140
2	1	4	3	20
3	2	1	3	50
4	1	2	5	90
	100	80	120	

- a) Find an initial basic feasible solution using the Northwest rule.
- b) Perform one iteration by entering the nonbasic variable with the most negative reduced cost into the basis.