Section 7.5: The Assignment Problem

In this section, we investigate the assignment problem- That is, given n jobs and n people, assign every job to a unique person. Typically, there are either costs or time involved, and we would want to make the assignments in such a way as to minimize this quantity.

Let's be more specific with the example from the text: MachineCo has 4 different machines, and each machine can do one of four jobs. We know the time involved for each machine and each job. Make assignments to minimize the total time involved.

Since we're minimizing time, can we frame this as a linear program? In fact we can, but it is probably easier to think of it as a tranportation problem (and through that, formulate the LP). For this to be put into a transportation problem, we want to think about each machine as a "supply" and each job as a "demand":

- Each machine as having a supply of 1.
- Each job has a demand of 1.
- Each machine, job pair has a cost.

This gives us the transportation tableau:

	J_1	J_2	J_3	J_4	
M_1	14	5	8	7	1
M_2	2	12	6	5	1
M_3	7	8	3	9	1
M_4	2	4	6	10	1
Demand	1	1	1	1	4

From this, we could solve it as a transportation problem or as a linear program. However, we can also take advantage of the form of the problem and put together an algorithm that takes advantage of it- this is the Hungarian Algorithm.

The Hungarian Algorithm

The Hungarian Algorithm is an algorithm designed to solve the assignment problem. We'll summarize it, but let's try the MachineCo problem as an example of how this algorithm will work.

First, we build a table with just our costs (or in this case, our times):

	J_1	J_2	J_3	J_4	min
M_1	14	5	8	7	
M_2	2	12	6	5	
M_3	7	8	3	9	
M_4	2	4	6	10	

Looking at this table, it is clear that no matter what job is assigned to machine 1, it will take at least 5 minutes. Similarly, in row 2, machine 2 takes at least 2 minutes. And to finish this off, machine 3 will take at least 3 minutes and machine 4, two minutes. We will take these minimum times out of each row (subtracting), and we'll end up with the following table. The sum of the times is given as 12 minutes- the minimum so far.

	J_1	J_2	J_3	J_4	min
M_1	9	0	3	2	5
M_2	0	10	4	3	2
M_3	4	5	0	6	3
M_4	0	2	4	8	2
					12

But wait- Looking down each row, it looks like no matter which machine job 4 is assigned to, it will take an additional 2 minutes. Therefore, the minimum of each column is written down at the bottom, and the columns are subtracted (in this case, only column 4). We add the two to 12 and get 14. The interpretation of this is that, no matter how the assignments are made, it will take at least 14 minutes as the minimum.

	J_1	J_2	J_3	J_4	min
M_1	9	0	3	0	5
M_2	0	10	4	1	2
M_3	4	5	0	4	3
M_4	0	2	4	6	2
	0	0	0	2	14

We would attain our minimum if we were able to make "0" assignments for each machine and job pair, but there aren't enough zeros yet.
We will now try to draw the minimum number of lines (horizontal or vertical only) we need to cover all the zeros currently showing. In this case, it takes three lines.

	J_1	J_2	J_3	J_4	min
M_1	~~9~~	~~0~~	~~3~~	~~0~~	5
M_2	0	10	4	1	2
M_3	4	5	0	4	3
M_4	0	2	4	6	2
	0	0	0	2	14

From the cells that remain uncovered, the minimum extra time that will be taken is 1 unit. Therefore, we'll subtract that from rows 2, 3, 4 (add to the right-most column).

	J_1	J_2	J_3	J_4	min
M_1	~~9~~	~~0~~	~~3~~	~~0~~	5
M_2	−1	9	3	0	3
M_3	3	4	−1	3	4
M_4	−1	1	3	5	3
	0	0	0	2	17

Unfortunately, subtracting one gave us some negative time cells. Add 1 back into columns 1 and 4 (bringing the -1 down to the bottom as our "minimum"). Notice this also takes our minimum time to 15.

	J_1	J_2	J_3	J_4	min
M_1	10	0	4	0	5
M_2	0	9	4	0	3
M_3	4	4	0	3	4
M_4	0	1	4	5	3
	-1	0	-1	2	15

What we would like to do now, is to consider what the end result was in moving from our initial table to our final table after adding and subtracting that minimum value:

	J_1	J_2	J_3	J_4	min
M_1	9	0	3	0	5
M_2	0	10	4	1	2
M_3	4	5	0	4	3
M_4	0	2	4	6	2
	0	0	0	2	14

$\Rightarrow$

	J_1	J_2	J_3	J_4	min
M_1	10	0	4	0	5
M_2	0	9	4	0	3
M_3	4	4	0	3	4
M_4	0	1	4	5	3
	-1	0	-1	2	15

Do you see that pattern? We subtract one from all cells not covered, add one to each cell containing an intersection, and leave the other cells.

	J_1	J_2	J_3	J_4	min
M_1	+1	0	+1	0	
M_2	0	-1	0	-1	
M_3	0	-1	0	-1	
M_4	0	-1	0	-1	

Now consider the array that remains. We are now able to assign zeros to a machine, job pairing, which means we'll attain the minimum cost.

	J_1	J_2	J_3	J_4	min
M_1	10	0	4	0	5
M_2	0	9	4	0	3
M_3	4	4	0	3	4
M_4	0	1	4	5	3
	-1	0	-1	2	15

As an example pairing, consider: (M_1, J_2), (M_2, J_4), (M_3, J_3), and (M_4, J_1). Then the time is going to be $5 + 5 + 3 + 2 = 15$, which is our minimum.

A Second Example

Let's look at a big longer example. See if you can do each step before looking at the answers.

Suppose we want to assign 4 chores to 4 people. The times for each person are given below. We want to assign one person per job in such a way as to minimize the time it takes. (Think of the jobs as either occurring one after another, or in terms of paying each person for each job).

	J_1	J_2	J_3	J_4
P_1	20	22	14	24
P_2	20	19	12	20
P_3	13	10	18	16
P_4	22	23	9	28

$\rightarrow$

	J_1	J_2	J_3	J_4	min
P_1	6	8	0	10	14
P_2	8	7	0	8	12
P_3	3	0	8	6	10
P_4	13	14	0	19	9
					45

$\rightarrow$

	J_1	J_2	J_3	J_4	min
P_1	3	8	0	4	14
P_2	5	7	0	2	12
P_3	0	0	8	0	10
P_4	10	14	0	13	9
	3	0	0	6	54

Now we see it takes at least 54 minutes no matter who is assigned to what. We would attain that minimum if we can assign a "0" to each person, job pairing. Unfortunately, there's no way that we can do that (yet). Use the minimum number of lines to cover all the zeros. In this case, it only takes two.

3	8	0	4
5	7	0	2
~~0~~	~~0~~	~~8~~	~~0~~
10	14	0	13

The **minimum amount** of the uncovered squares is 2. The operation we perform is given below:

3	8	0	4
5	7	0	2
0	0	8	0
10	14	0	13

$+$

-2	-2		-2
-2	-2		-2
		$+2$	
-2	-2		-2

$=$

1	6	0	2
3	5	0	0
0	0	10	0
8	12	0	11

I should mention that when you do these in practice, we don't typically keep track of the "current minimum"- We can just compute the minimum at the end. Sometimes it is beneficial to track it if we want to track the solution to the dual (these minimum values perhaps?).

Now we repeat- What is the minimum number of (horizontal or vertical) lines required to cover all the zeros? If less than 4, we have to continue. In this case, we have three lines. The remaining minimum value is 1.

1	6	0	2
3	5	0	0
~~0~~	~~0~~	~~10~~	~~0~~
8	12	0	11

And now we have our last step. We assign the jobs to the red zeros in the array to the right, and that gives us the minimum time (59 units). You might also note that we cannot cover the zeros with less than 4 lines.

0	5	0	2
2	4	0	0
0	0	11	1
7	11	0	11

$\Rightarrow$

0	5	0	2
2	4	0	0
0	0	11	1
7	11	0	11

The Hungarian Method for Solving the Assignment Problem

We're ready to state the Hungarian method now that we've seen a couple of examples.

- Initialize the algorithm:
  - Subtract the lowest row value from each row.
  - For each column, subtract the lowest value. Steps 1 and 2 create zeros to start the algorithm off.
    Side Remark: This also initializes the solution to the dual.
- Repeat:
  - Cover the zeros with as few lines as possible. If the number of lines is less than n (in this case, 4), then we are not optimal. If equal to n, stop.
  - Find the minimum number in the cells that are uncovered, and subtract that value from all uncovered cells.
  - Add that number to the cells wherever two lines intersect.

If the number of lines required to cover all of the zeros is n, then we are able to assign one zero to each person/job pair as follows:

Assignment of Zeros Sub-Algorithm

We do have to be a bit careful when assigning the zeros- We want to be sure that we don't end up with a sub-optimal solution due to "bad decisions" up front. For example, in the first example, what happens if we assign the jobs, in order:

$$\begin{matrix} 10 & 0 & 4 & 0 \\ 0 & 9 & 4 & 0 \\ 4 & 4 & 0 & 3 \\ 0 & 1 & 4 & 5 \end{matrix} \rightarrow \begin{matrix} 10 & 0 & 4 & 0 \\ 0 & 9 & 4 & 0 \\ 4 & 4 & 0 & 3 \\ 0 & 1 & 4 & 5 \end{matrix} \rightarrow \begin{matrix} 10 & 0 & 4 & 0 \\ 0 & 9 & 4 & 0 \\ 4 & 4 & 0 & 3 \\ 0 & 1 & 4 & 5 \end{matrix} \rightarrow ??$$

So there is a little algorithm that will help us from getting trapped like this:

- Check rows first. If a row has exactly one zero, assign it (otherwise, leave the row for now). If a cell has been assigned, close the row and column.
- Check open columns. If a column has exactly one zero, assign it and close the row/column.

Of course, there are special arrays that could be problematic if there are many zeros. Here's another quick example:

$$\begin{matrix} 3 & 1 & 1 & 4 \\ 4 & 2 & 2 & 5 \\ 5 & 3 & 4 & 8 \\ 4 & 2 & 5 & 9 \end{matrix} \Rightarrow \cdots \Rightarrow \begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 3 & 4 \end{matrix}$$

There are many ways to assign the jobs- Notice that a couple of them may get you into trouble (make cell $(1,1)$ and $(2,1)$ assignments).

A More Extended Example

In this example, we'll have a 5×5 to see how the algorithm works on a larger problem. You should write down this initial table and see if you can perform the algorithm, and check back here after you're done. In this case, we won't track the minimums.

The first step shown is to remove the smallest values in each row, the second step is the minimum of each column.

11	7	10	17	10
13	21	7	11	13
13	13	15	13	14
18	10	13	16	14
12	8	16	19	10

$\Rightarrow$

4	0	3	10	3
6	14	0	4	6
0	0	2	0	1
8	0	3	6	4
4	0	8	11	2

$\Rightarrow$

4	0	3	10	2
6	14	0	4	5
0	0	2	0	0
8	0	3	6	3
4	0	8	11	1

Now three lines will cover all the zeros. We get the minimum value of the remaining cells (which is 1). Subtract from all uncovered cells, add in cells that have intersections.

4	0	3	10	2
6	14	0	4	5
0	0	2	0	0
8	0	3	6	3
4	0	8	11	1

$\Rightarrow$

3	0	2	9	1
6	15	0	4	5
0	1	2	0	0
7	0	2	5	2
3	0	7	10	0

$\Rightarrow$

3	0	2	9	1
6	15	0	4	5
0	1	2	0	0
7	0	2	5	2
3	0	7	10	0

Now the minimum value is 1 again, so repeat. The minimum value of the remaining uncovered cells is now 2, and perform the algorithm again. We now have an optimal assignment.

2	0	1	8	0
6	16	0	4	5
0	2	2	0	0
6	0	1	4	1
3	1	7	10	0

$\Rightarrow$

2	0	1	8	0
6	16	0	4	5
0	2	2	0	0
6	0	1	4	1
3	1	7	10	0

$\Rightarrow$

0	0	1	6	0
4	16	0	2	5
0	4	4	0	2
4	0	1	2	1
1	1	7	8	0

And we note that the final cost is the sum of these c_{ij}:

$$w = 11 + 7 + 13 + 10 + 10 = 51$$

Extensions

- The first extension: What if someone can take on more than one job?

  You would have one row for the first job assignment, and a second row for the second job assignment.

- What if we have fewer jobs than people?

  You would create a dummy job.

- What if some people can not do certain jobs?

  Replace the cost with M, a very large number.

- Is it possible to convert a transportation problem into an assignment problem?

  Here's an example (Problem 7 from the text). The main idea- Create a supply point for each unit of supply and a demand point for each unit of demand.

  Here's the original tableau (costs shown):

			Supply
	3	1	2
	2	3	3
Demand	1	4	

  We have 5 units of supply and 5 units of demand, so our new table is 5×5, partitioned by (1,4) along the bottom for demand, and (2,3) along the side from supply.

3	1	1	1	1	1
3	1	1	1	1	1
2	3	3	3	3	1
2	3	3	3	3	1
2	3	3	3	3	1
1	1	1	1	1	

  These would be the assignment costs.

- What if we want to maximize the sum of the costs instead of minimize?

  Negate all costs, then minimize. Here's an example:

$$\begin{array}{cccc} 7 & 5 & 8 & 2 \\ 7 & 8 & 9 & 4 \\ 3 & 5 & 7 & 9 \\ 5 & 5 & 6 & 7 \end{array} \quad \rightarrow \quad \begin{array}{cccc} -7 & -5 & -8 & -2 \\ -7 & -8 & -9 & -4 \\ -3 & -5 & -7 & -9 \\ -5 & -5 & -6 & -7 \end{array}$$

  Now performing the first step of the algorithm will make this look "normal":

$$\begin{array}{cccc|c} 1 & 3 & 0 & 6 & (-8) \\ 2 & 1 & 0 & 5 & (-9) \\ 6 & 4 & 2 & 0 & (-9) \\ 2 & 2 & 1 & 0 & (-7) \\ \hline * & * & * & * & (-33) \end{array} \quad \rightarrow \quad \begin{array}{cccc|c} \cancel{0} & \cancel{2} & \cancel{0} & \cancel{6} & (-8) \\ \cancel{1} & \cancel{0} & \cancel{0} & \cancel{5} & (-9) \\ 5 & 3 & 2 & 0 & (-9) \\ 1 & 1 & 1 & 0 & (-7) \\ \hline (1) & (1) & 0 & 0 & (-31) \end{array}$$

0	2	0	6	(-8)
1	0	0	5	(-9)
4	2	1	0	(-8)
0	0	0	0	(-6)
(1)	(1)	0	(-1)	(-30)

For assignments,

0	2	0	6	(-8)
1	0	0	5	(-9)
4	2	1	0	(-8)
0	0	0	0	(-6)
(1)	(1)	0	(-1)	(-30)