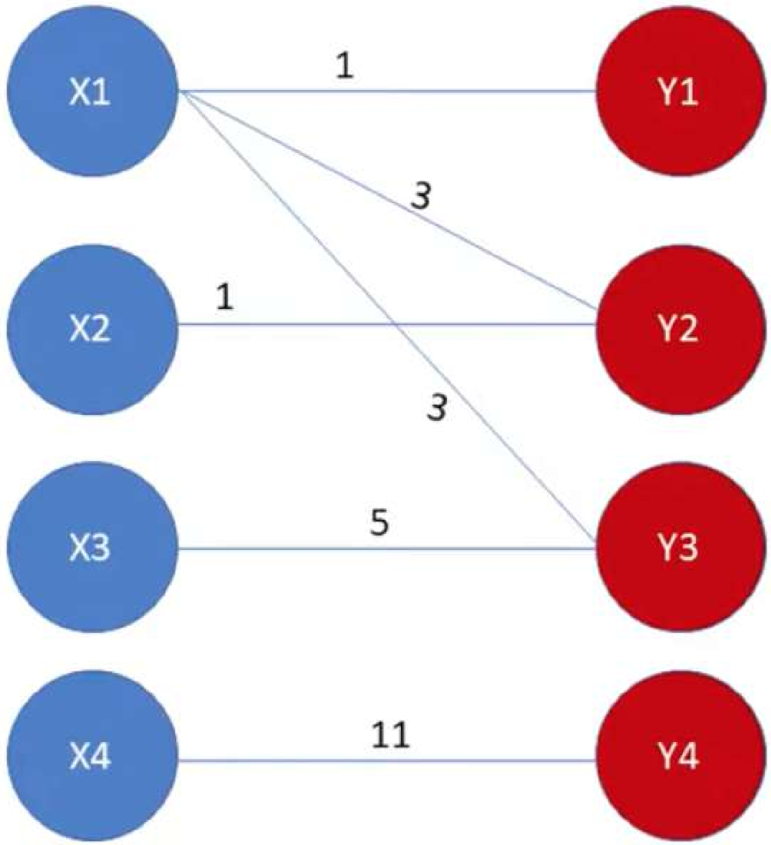
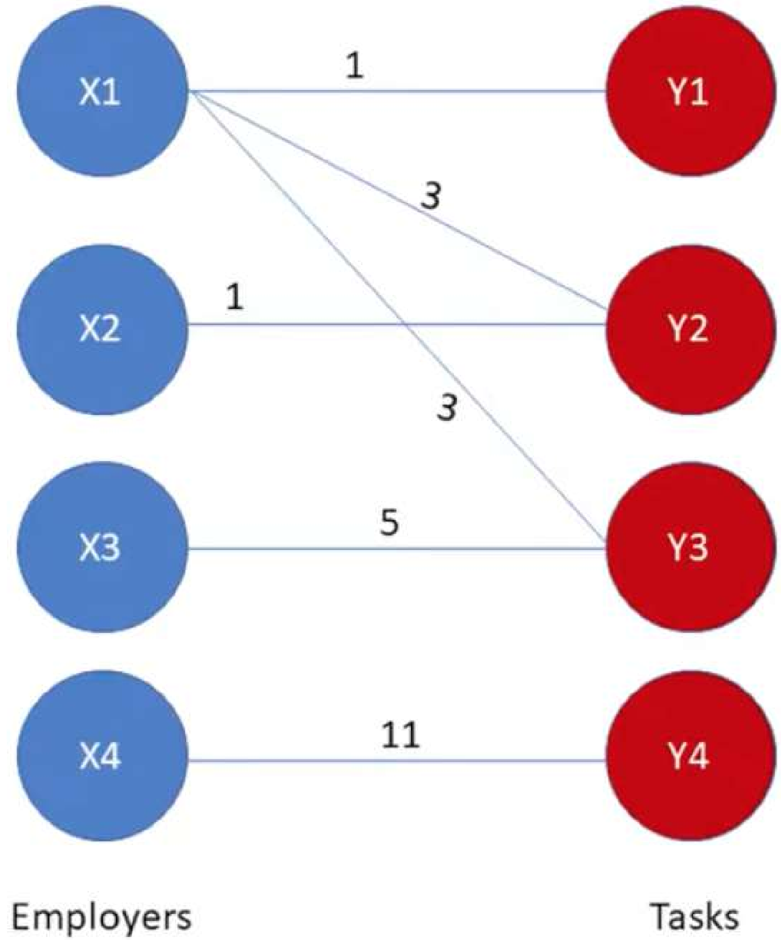
Matching in Weighted Bipartite Graphs: The Hungarian Algorithm

Exercise Lecture

Bipartite Graphs



Example



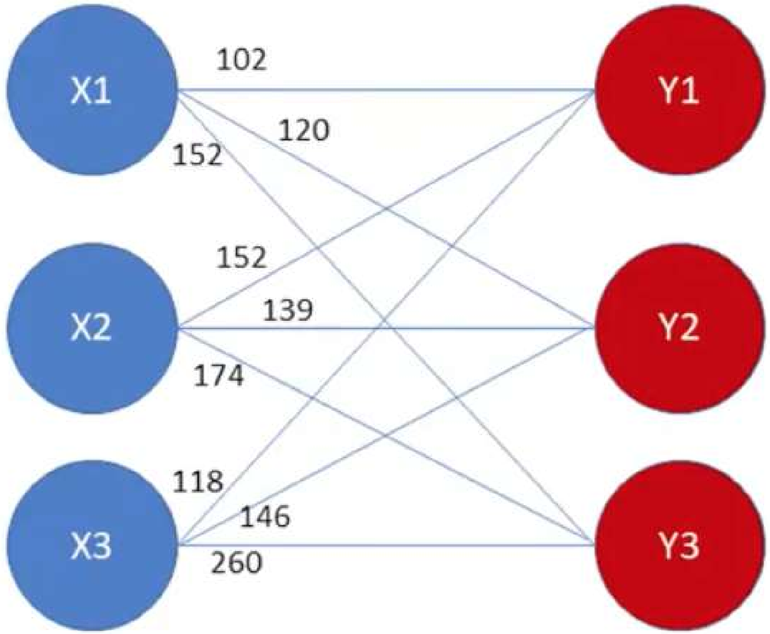
Example

How should I assign employers to tasks in order to maximize the overall performance?

Two Approaches:

- **Hungarian Algorithm using an adjacency matrix**
  - Suitable for **dense graphs**
  - The algorithm returns **the minimal cost matching**
  - Many libraries and functions available online (e.g., scipy.optimize.linear_sum_assignment)
- **Hungarian Algorithm using a graph**
  - Suitable for **sparse graphs**
  - Some libraries available online (e.g., NetworkX)

Adjacency Matrix



	Y1	Y2	Y3
X1	102	120	152
X2	152	139	174
X3	118	146	260

Hungarian Algorithm: Key Ideas

- The algorithm returns the **minimal cost matching**
- If a number is added to all of the entries of anyone row or column of a cost matrix, then an optimal assignment for the resulting cost matrix **is also an optimal assignment for the original cost matrix.**
- We can compute the maximum matching by minimizing the loss instead of the initial weights
- If the original graph is not balanced we can dummy variables with the maximum cost (zero if we are maximizing) as value.

	Y1	Y2	Y3
X1	102	120	152
X2	152	139	174
X3	118	146	260

Hungarian Algorithm: Steps

1. Subtract the smallest entry in each row from all the other entries in the row.
2. Subtract the smallest entry in each column from all the other entries in the column.
3. Draw lines through the row and columns that have the 0 entries such that the fewest lines possible are drawn.
4. If there are N lines drawn, an optimal assignment of zeros is possible and the algorithm is finished. Otherwise, go to the next step.
5. Find the smallest entry not covered by any line. Subtract this entry from each row that is not crossed out, and then add it to each column that **is** crossed out. Then, go back to Step 3.

	Y1	Y2	Y3
X1	102	120	152
X2	152	139	174
X3	118	146	260

Step 1

1. Subtract the smallest entry in each row from all the other entries in the row.

	Y1	Y2	Y3
X1	102	120	152
X2	152	139	174
X3	118	146	260

	Y1	Y2	Y3
X1	0	18	50
X2	13	0	35
X3	0	28	142

Step 2

- Subtract the smallest entry in each column from all the other entries in the column.

	Y1	Y2	Y3
X1	0	18	50
X2	13	0	35
X3	0	28	142

	Y1	Y2	Y3
X1	0	18	15
X2	13	0	0
X3	0	28	107

Step 3

- Draw lines through the row and columns that have the 0 entries such that the fewest lines possible are drawn.

	Y1	Y2	Y3
X1	0	18	15
X2	13	0	0
X3	0	28	107

Step 3

- Draw lines through the row and columns that have the 0 entries such that the fewest lines possible are drawn.

	Y1	Y2	Y3
X1	0	18	15
X2	13	0	0
X3	0	28	107

Step 3: Implementation Details

	Y1	Y2	Y3	
X1	0	18	15	
X2	13	0	0	
X3	0	28	107	✓

1. Find a start assignment covering as many tasks (y) as possible.

2. Mark all rows having no assignment (row 3).

Step 3: Implementation Details

	Y1	Y2	Y3	
	✓			
X1	0	18	15	
X2	13	0	0	
X3	0	28	107	✓

1. Find a start assignment covering as many tasks (y) as possible.
2. Mark all rows having no assignment (row 3).
3. Mark all (unmarked) columns having zeros in newly marked row(s) (col 1).

.

Step 3: Implementation Details

	Y1	Y2	Y3
X1	0	18	15
X2	13	0	0
X3	0	28	107

1. Find a start assignment covering as many tasks (y) as possible.
2. Mark all rows having no assignment (row 3).
3. Mark all (unmarked) columns having zeros in newly marked row(s) (col 1).
4. Mark all rows having assignments in newly marked columns (row 1).

Step 3: Implementation Details

	Y1	Y2	Y3
X1	0	18	15
X2	13	0	0
X3	0	28	107

1. Find a start assignment covering as many tasks (y) as possible.
2. Mark all rows having no assignment (row 3).
3. Mark all (unmarked) columns having zeros in newly marked row(s) (col 1).
4. Mark all rows having assignments in newly marked columns.
5. Repeat for all non-assigned rows.
6. Finally, select **marked columns** and **unmarked rows**.

Step 4

- If there are N lines drawn, an optimal assignment of zeros is possible and the algorithm is finished. Otherwise, go to the next step.

	Y1	Y2	Y3
X1	0	18	15
X2	13	0	0
X3	0	28	107

The number of lines is lower than 3. We have to go to step 5!

Step 5

- Find the smallest entry not covered by any line. Subtract this entry from each row that is not crossed out, and then add it to each column that **is** crossed out.

	Y1	Y2	Y3
X1	0	18	15
X2	13	0	0
X3	0	28	107

Step 5

- Find the smallest entry not covered by any line. Subtract this entry from each row that is not crossed out, and then add it to each column that **is** crossed out.

	Y1	Y2	Y3
X1	0	18	15
X2	13	0	0
X3	0	28	107

→

	Y1	Y2	Y3
X1	0	3	0
X2	28	0	0
X3	0	13	92

Go to Step 3

Step 3

- Draw lines through the row and columns that have the 0 entries such that the fewest lines possible are drawn.

	Y1	Y2	Y3
X1	0	3	0
X2	28	0	0
X3	0	13	92

The number of lines is equal to 3. We can find an optimal zeros assignment!

Final Assignment: Implementation Details

1. Examine the rows successively until a row-wise exactly single zero is found; mark this zero by '1' to make the assignment.

	Y1	Y2	Y3
X1	0	3	0
X2	28	0	0
X3	1	13	92

Final Assignment: Implementation Details

	Y1	Y2	Y3
X1	1	3	0
X2	28	1	1
X3	1	13	92

1. Examine the rows successively until a row-wise exactly single zero is found; mark this zero by '1' to make the assignment.

2. Mark all zeroes lying in the column of the marked zero.

3. Do the same procedure for the columns also.

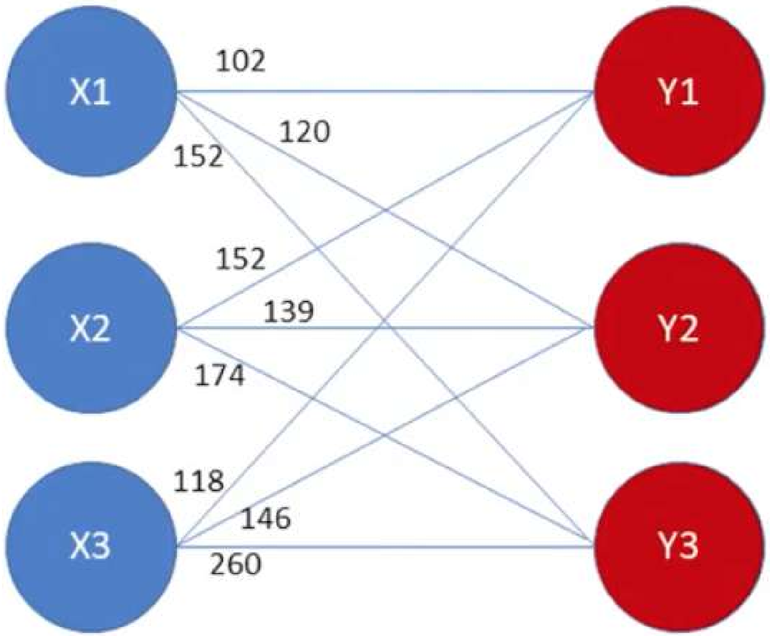
4. Repeat until there are not rows and columns with single zeros.

Final Assignment: Implementation Details

	Y1	Y2	Y3
X1	1	3	1
X2	28	1	1
X3	1	13	92

1. Examine the rows successively until a row-wise exactly single zero is found; mark this zero by ‘1’ to make the assignment.
2. Mark all zeroes lying in the column of the marked zero.
3. Do the same procedure for the columns also.
4. Repeat until there are not rows and columns with single zeros.
5. If there lies more than one of the unmarked zeroes in any column or row, then mark ‘1’‘one of the unmarked zeroes arbitrarily and mark a cross in the cells of remaining zeroes in its row and column.

Final Assignment



Final Assignment