Lecture 8

Lecture 8

Previously in Opt 2 …

The Assignment Problem

- Input:
  - A set of n workers: W = {1, 2, .., n}
  - A set of n tasks: T = {1, 2, …, n}
  - For each pair of worker i and task j:
    t_{ij} = time (or cost) for worker i to finish task j
- Objective:
  - To minimize the total time (or cost) for finishing all n tasks
- Constraint:
  - Each worker is assigned to do one task
  - Each task is assigned to one worker

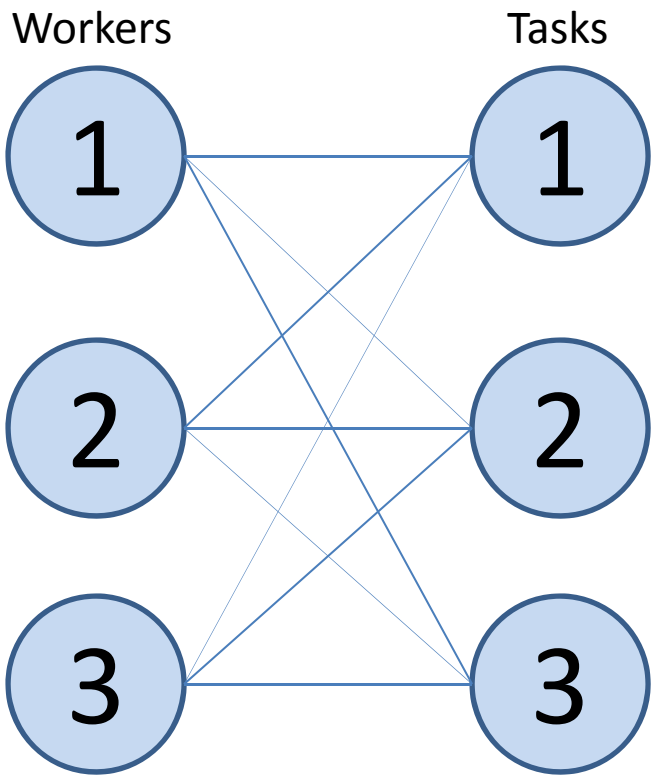
The Assignment Problem: Example 0

n = 3

With cost table:

(Row i = worker i; Column j = task j)

1	0	3
0	4	0
0	2	0



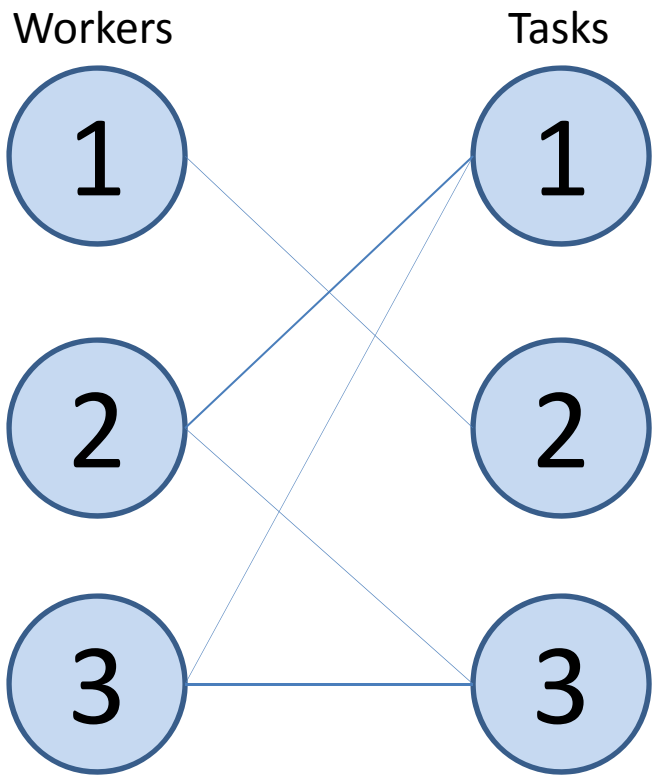
The Assignment Problem: Example 0

n = 3

With cost table:

(Row i = worker i; Column j = task j)

1	0	3
0	4	0
0	2	0



Try to use only edges with zero costs in our assignment.

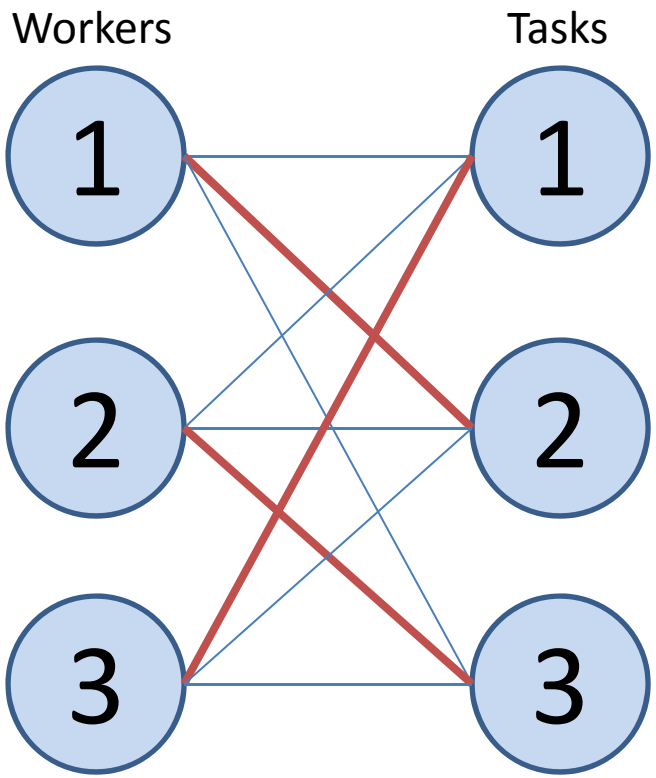
The Assignment Problem: Example 0

n = 3

With cost table:

(Row i = worker i; Column j = task j)

1	0	3
0	4	0
0	2	0



The assignment:
(1, 2), (2, 3), (3, 1)
is optimal.

The Assignment Problem: An observation

Observation 1

If $t_{ij} \geq 0$ for all pairs (i, j), and if we have an assignment of zero total cost,

then this assignment is optimal.

Moreover, each pairing in this assignment must have zero cost.

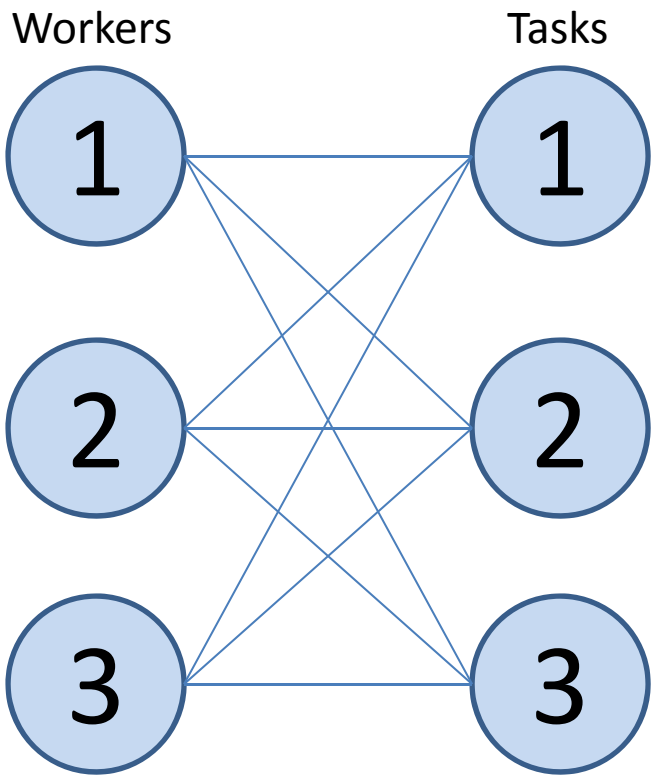
The Assignment Problem: Example 1

n = 3

With cost table:

(Row i = worker i; Column j = task j)

6	4	7
2	5	1
3	4	2



The Assignment Problem: Another observation

Claim 1

Suppose x denote an optimal assignment to an input with edge costs t_{ij}.

Suppose that we subtract α_i from each entry of row i (or β_j from each entry of column j).

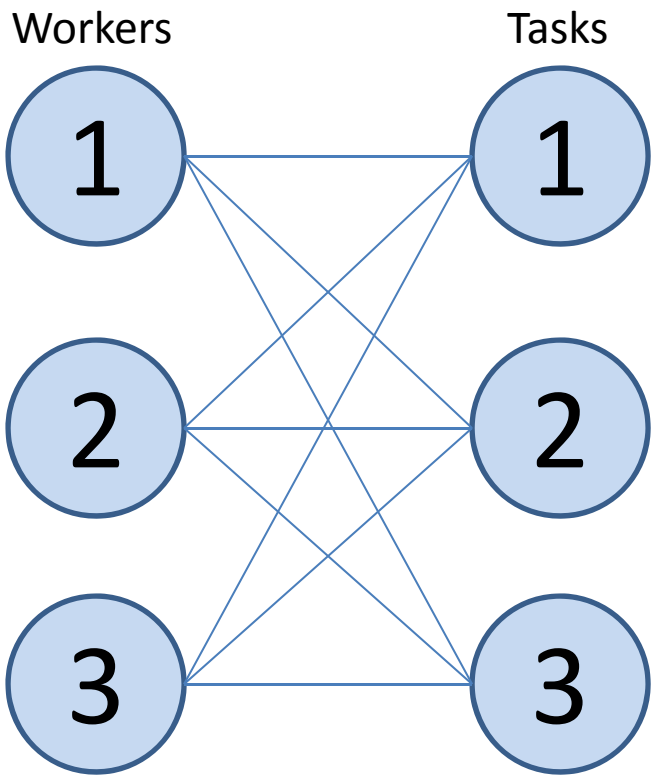
(i.e., subtract α_i from the cost of each edge adjacent to worker-node i, or β_j from the cost of each edge adjacent to task-node j, respectively)

Then x is still optimal for the modified input.

The Assignment Problem: Example 1

Subtract 4 from each entry in row 1

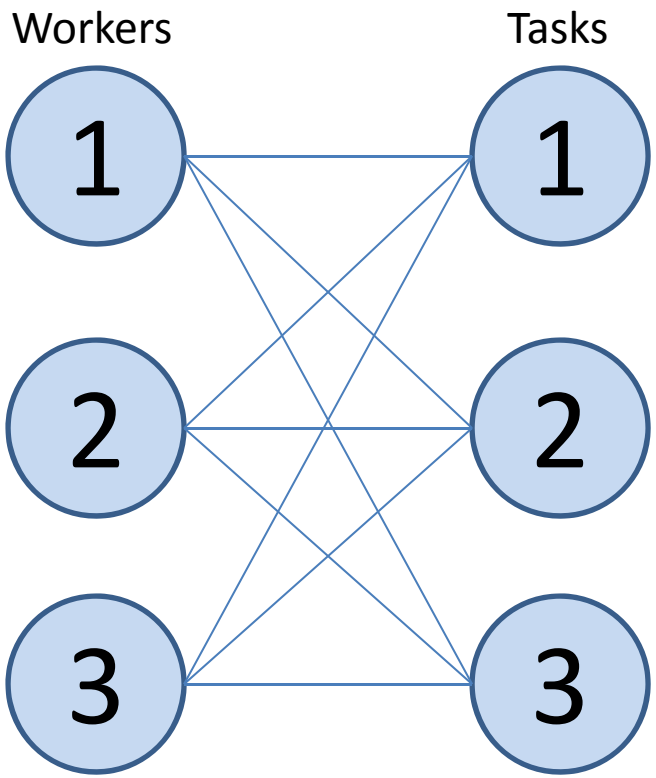
6	4	7
2	5	1
3	4	2



The Assignment Problem: Example 1

Subtract 4 from each entry in row 1

2	0	3
2	5	1
3	4	2



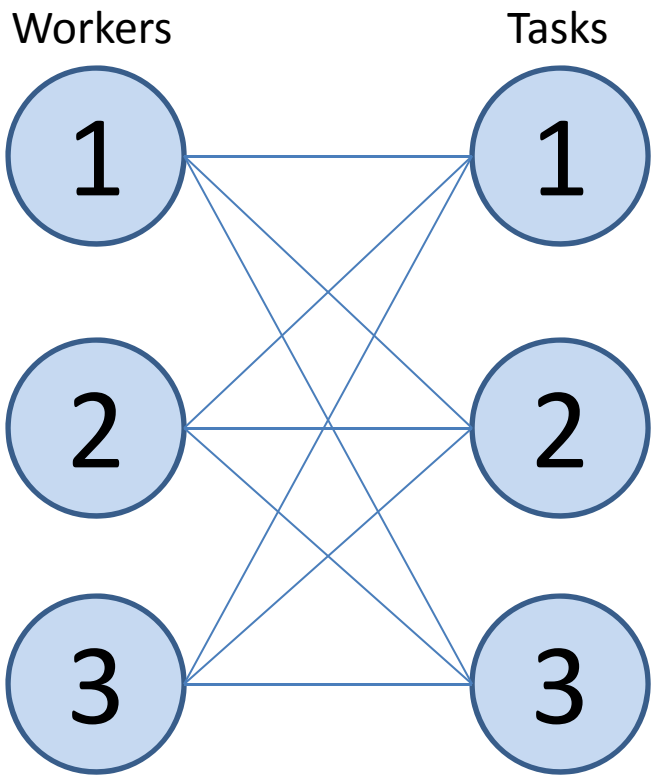
The Assignment Problem: Example 1

Subtract 4 from each entry in row 1

Subtract 1 from each entry in row 2

Subtract 2 from each entry in row 3

2	0	3
2	5	1
3	4	2



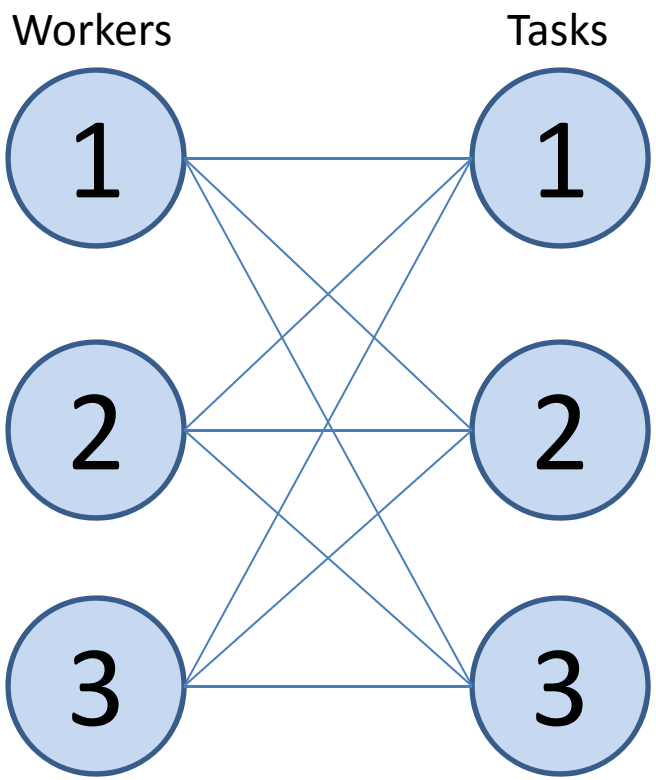
The Assignment Problem: Example 1

Subtract 4 from each entry in row 1

Subtract 1 from each entry in row 2

Subtract 2 from each entry in row 3

2	0	3
1	4	0
1	2	0



The Assignment Problem: Example 1

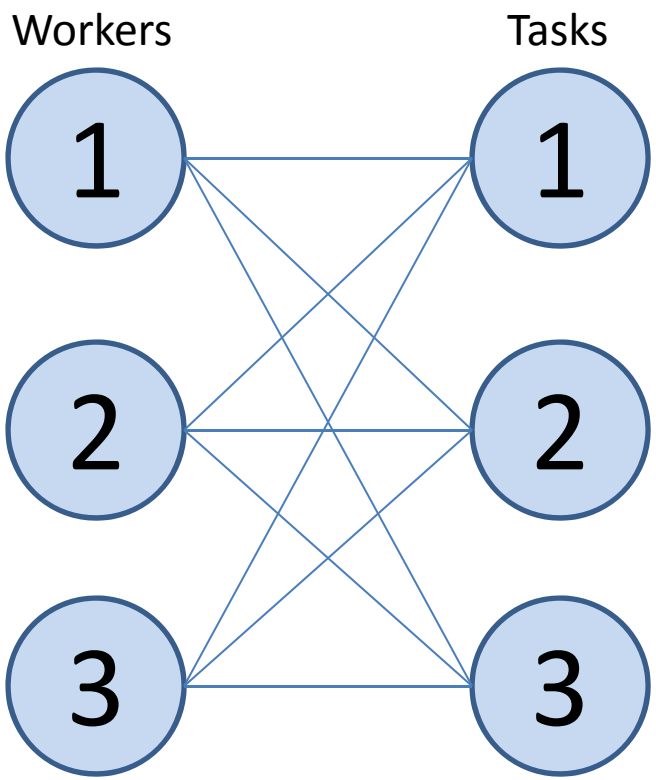
Subtract 4 from each entry in row 1

Subtract 1 from each entry in row 2

Subtract 2 from each entry in row 3

Subtract 1 from each entry in col 1

2	0	3
1	4	0
1	2	0



The Assignment Problem: Example 1

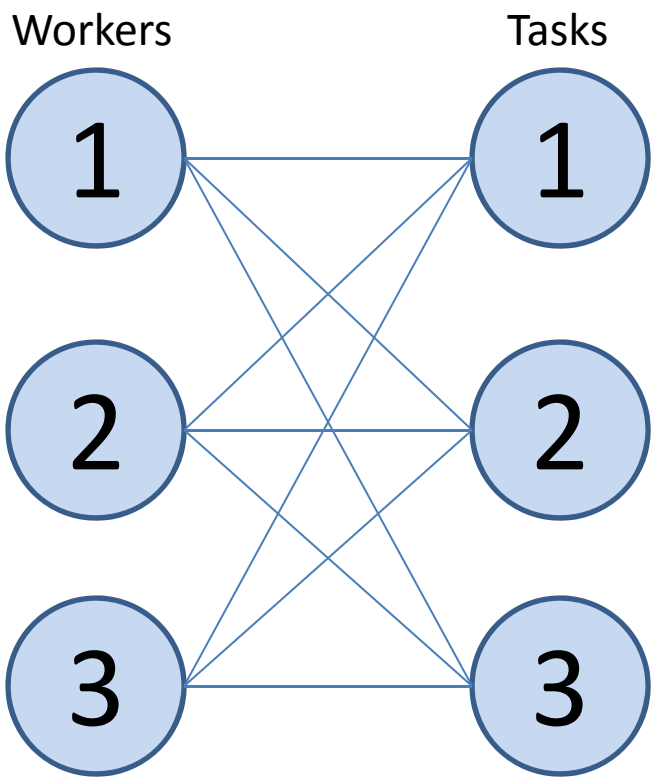
Subtract 4 from each entry in row 1

Subtract 1 from each entry in row 2

Subtract 2 from each entry in row 3

Subtract 1 from each entry in col 1

1	0	3
0	4	0
0	2	0

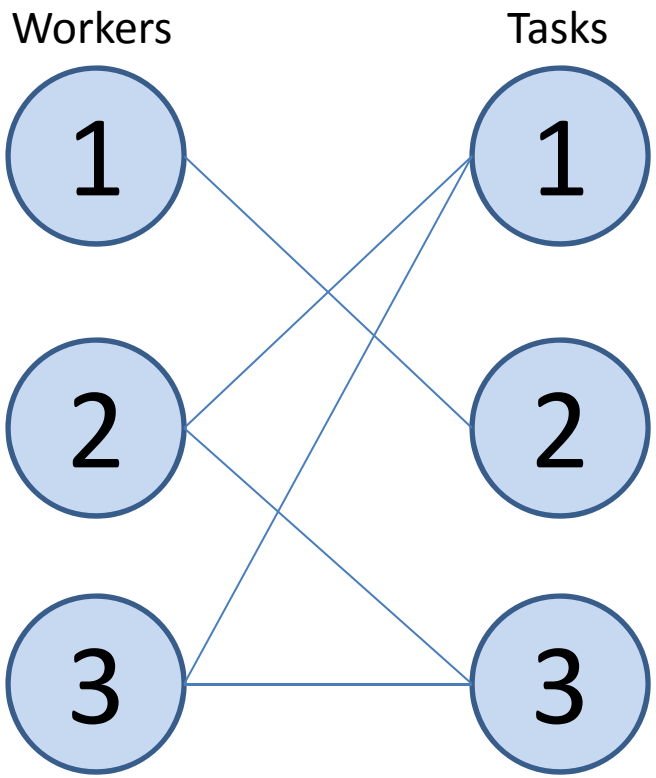


The Assignment Problem: Example 1

Then, find a pairing of workers to tasks that only uses edges of zero costs.

That is, find an ***all-zero-assignment***.

1	0	3
0	4	0
0	2	0



Displayed above are only the edges that have zero costs.

The Assignment Problem: Example 1

Then, find a pairing of workers to tasks that only uses edges of zero costs.

That is, find an ***all-zero-assignment***.

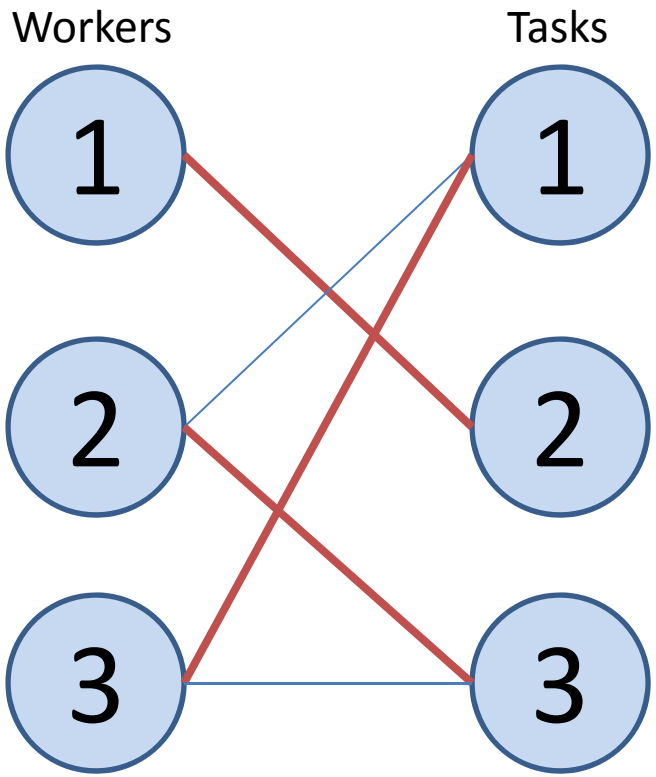
1	0	3
0	4	0
0	2	0

Definition

An *all-zero-assignment* is a feasible assignment of all n workers to n tasks, which uses only the edges that have costs equal to zero.

The Assignment Problem: Example 1

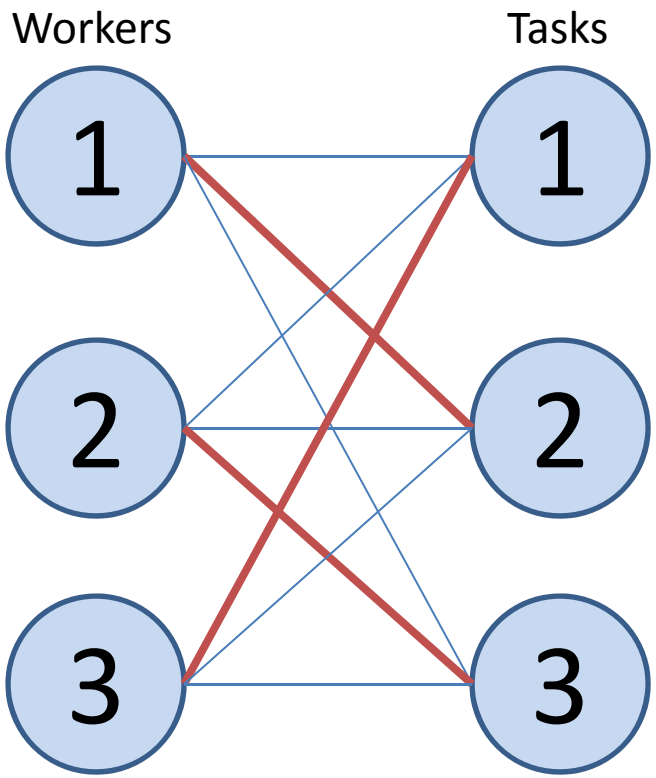
1	0	3
0	4	0
0	2	0



The assignment:
(1, 2), (2, 3), (3, 1)
is an ***all-zero-assignment***.

The Assignment Problem: Example 1

Use the assignment we just found in the original problem.



6	4	7
2	5	1
3	4	2

By Claim 1, the assignment:

(1, 2), (2, 3), (3, 1)

is also optimal for the original problem, with total cost = 4 + 1 + 3 = 8

The Assignment problem: A first algorithm

0. Set up a cost table. Assume all costs are ≥ 0.
1. For each row i = 1, …, n
   let α_i = the smallest entry in row i
   update: $t_{ij} \leftarrow t_{ij} - \alpha_i$ for all entries (i,j) in row i
2. For each col j = 1, …, n
   let β_j = the smallest entry in column j
   update: $t_{ij} \leftarrow t_{ij} - \beta_j$ for all entries (i,j) in col j
3. Check if there is an all-zero assignment
   If there is one, this assignment is optimal

The Assignment problem: A first algorithm

0. Set up a cost table. Assume all costs are ≥ 0.
1. For each row i = 1, …, n
   let α_i = the smallest entry in row i
   update: $t_{ij} \leftarrow t_{ij} - \alpha_i$ for all entries (i,j) in row i
2. For each col j = 1, …, n
   let β_j = the smallest entry in column j
   update: $t_{ij} \leftarrow t_{ij} - \beta_j$ for all entries (i,j) in col j
3. Check if there is an all-zero assignment
   If there is one, this assignment is optimal
   If there isn't one: …?

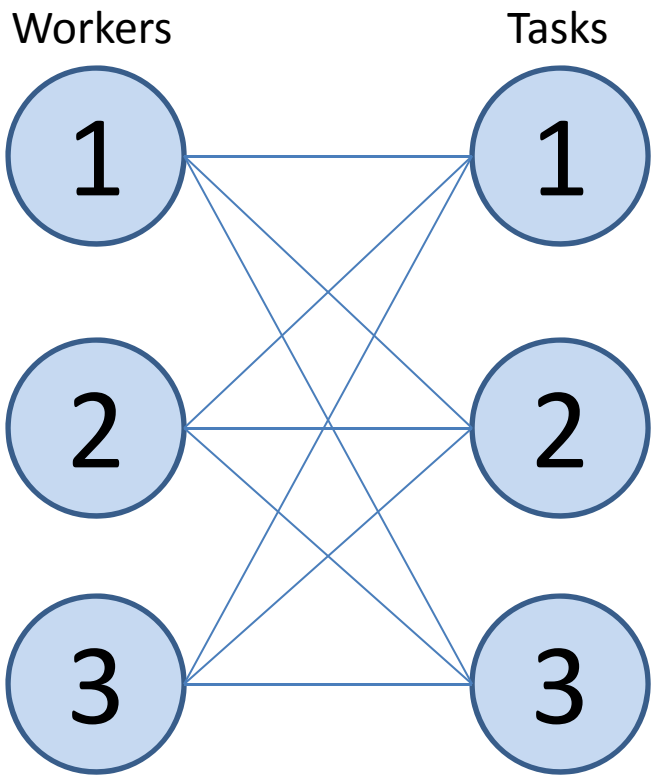
The Assignment Problem: Example 2

n = 3

With cost table:

(Row i = worker i; Column j = task j)

8	10	7
10	11	8
5	6	7



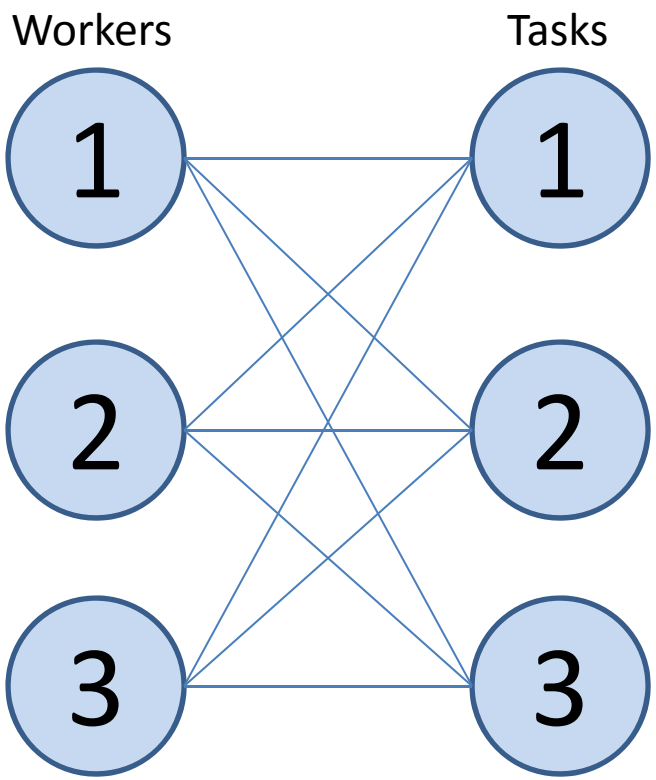
The Assignment Problem: Example 2

Subtract 7 from each entry in row 1

Subtract 8 from each entry in row 2

Subtract 5 from each entry in row3

8	10	7
10	11	8
5	6	7



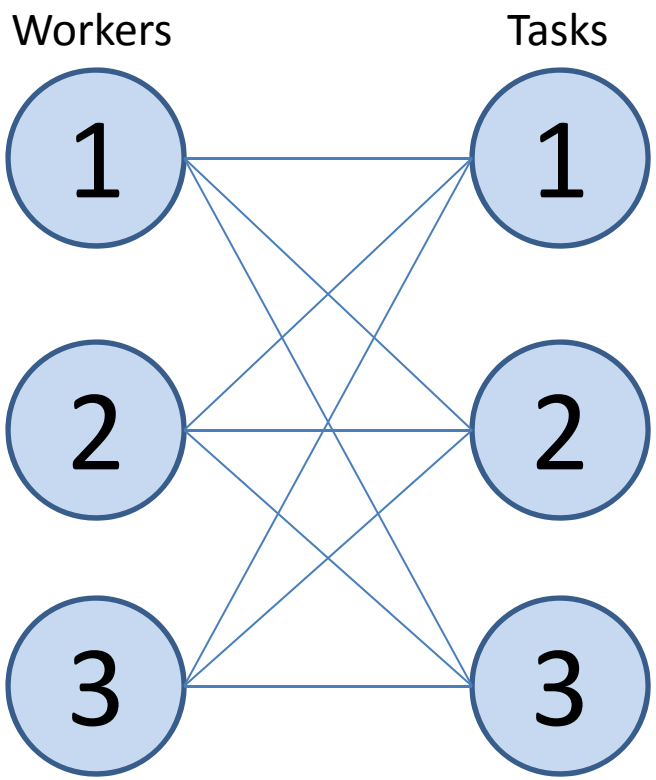
The Assignment Problem: Example 2

Subtract 7 from each entry in row 1

Subtract 8 from each entry in row 2

Subtract 5 from each entry in row3

1	3	0
2	3	0
0	1	2



The Assignment Problem: Example 2

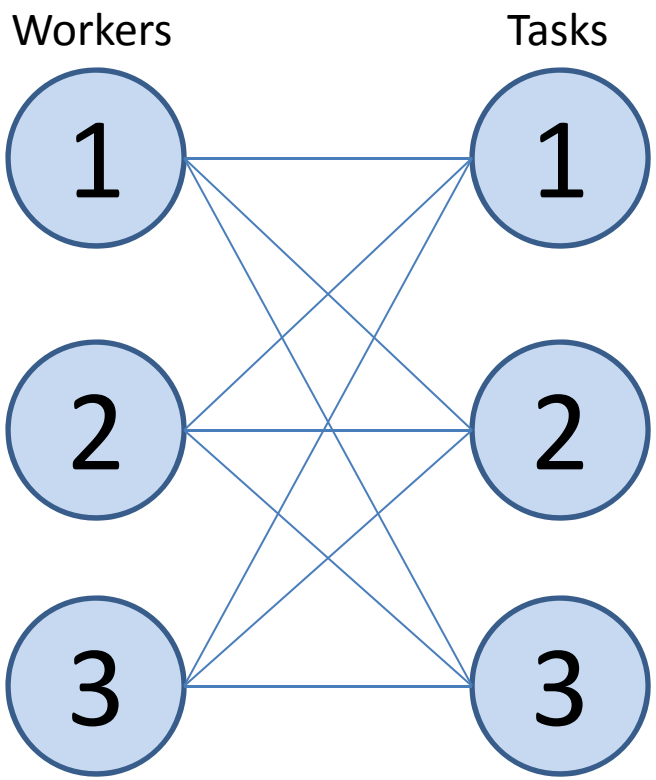
Subtract 7 from each entry in row 1

Subtract 8 from each entry in row 2

Subtract 5 from each entry in row3

Subtract 1 from each entry in col 2

1	3	0
2	3	0
0	1	2



The Assignment Problem: Example 2

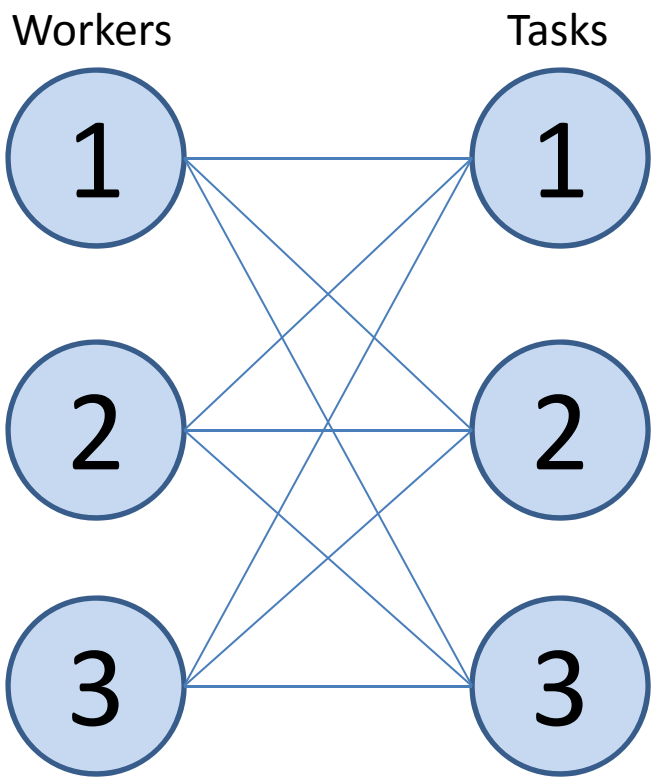
Subtract 7 from each entry in row 1

Subtract 8 from each entry in row 2

Subtract 5 from each entry in row3

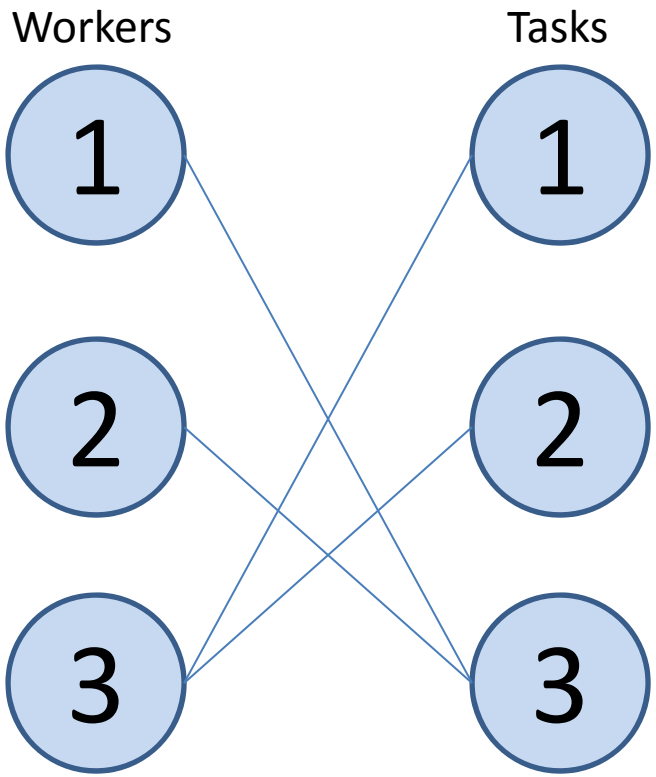
Subtract 1 from each entry in col 2

1	2	0
2	2	0
0	0	2



The Assignment Problem: Example 2

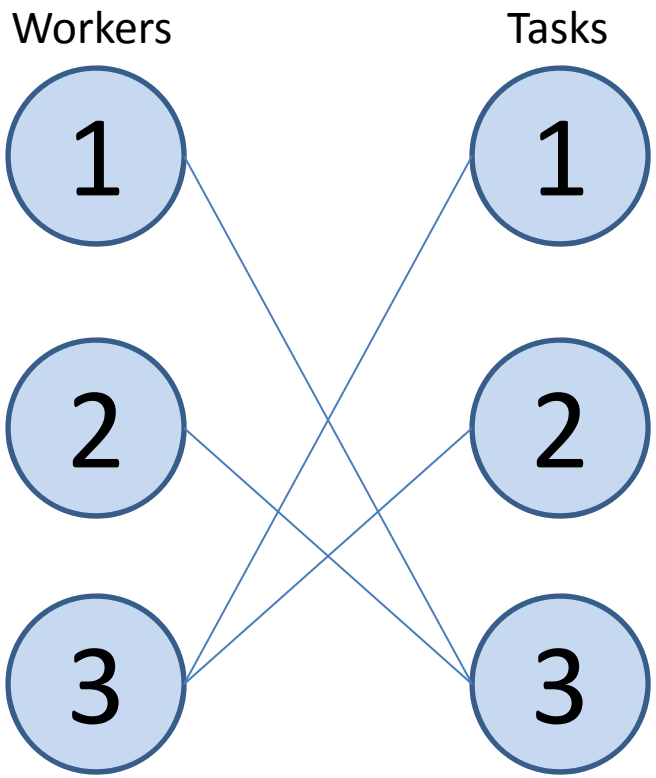
1	2	0
2	2	0
0	0	2



Displayed above are only the edges that have zero costs.

The Assignment Problem: Example 2

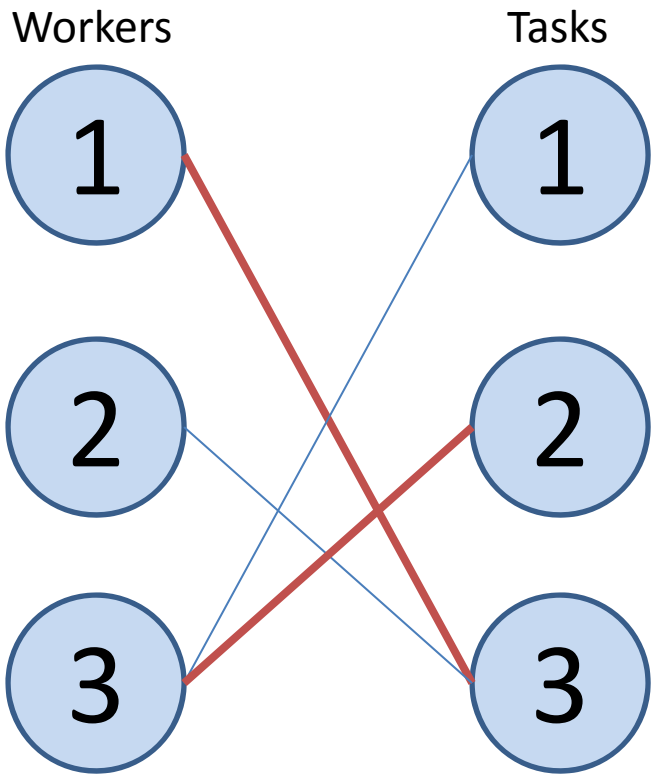
1	2	0
2	2	0
0	0	2



There is no all-zero-assignment:
Can form at most two pairings
using the edges with costs zero.

The Assignment Problem: Example 2

1	2	0
2	2	0
0	0	2



There is no all-zero-assignment:
Can form at most two pairings
using the edges with costs zero.

The Assignment problem: A first algorithm

0. Set up a cost table. Assume all costs are ≥ 0.
1. For each row i = 1, …, n
   let α_i = the smallest entry in row i
   update: $t_{ij} \leftarrow t_{ij} - \alpha_i$ for all entries (i,j) in row i
2. For each col j = 1, …, n
   let β_j = the smallest entry in column j
   update: $t_{ij} \leftarrow t_{ij} - \beta_j$ for all entries (i,j) in col j
3. Check if there is an all-zero assignment
   If there is one, this assignment is optimal
   If there isn't one: …?

The Assignment Problem: Towards the Hungarian Algorithm

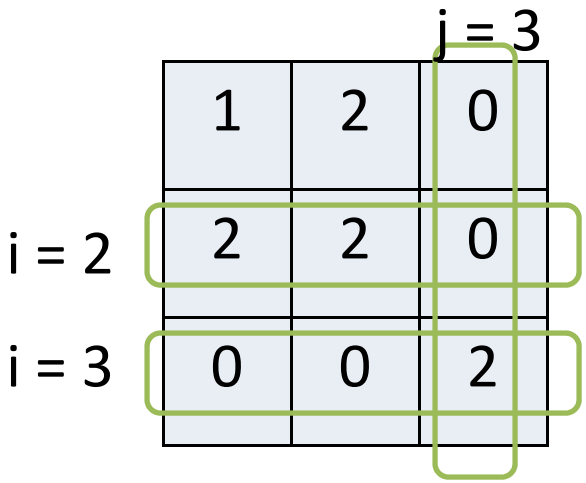
Definition

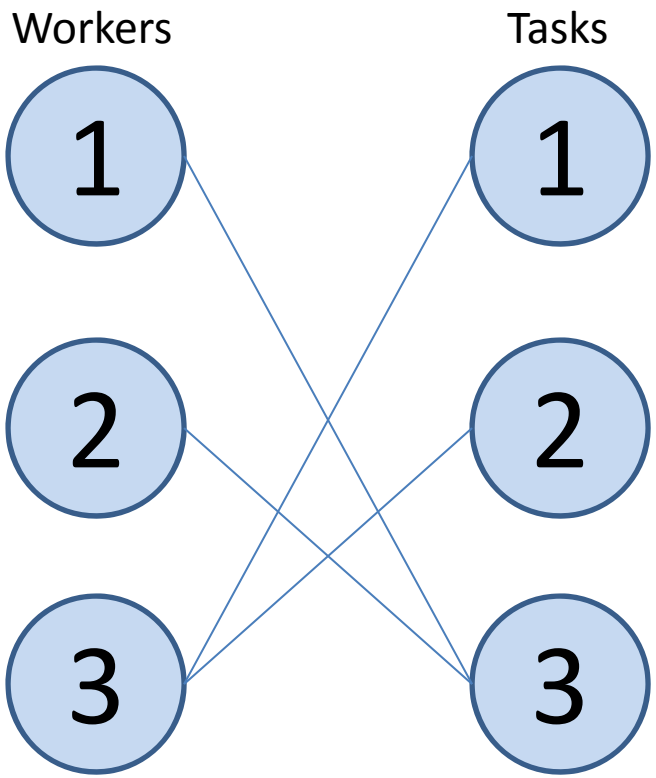
Given a cost table for the assignment problem,

a zero-cover is a collection of rows and columns such that each “0” entry in the table lies in either a row or a column in the zero-cover.

The Assignment Problem: Example 2

Example of a zero-cover

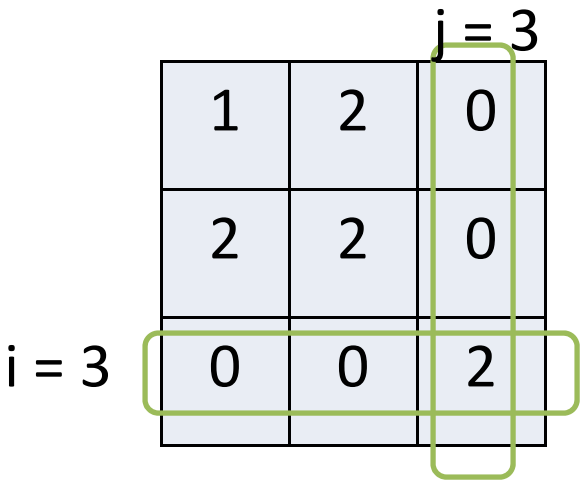


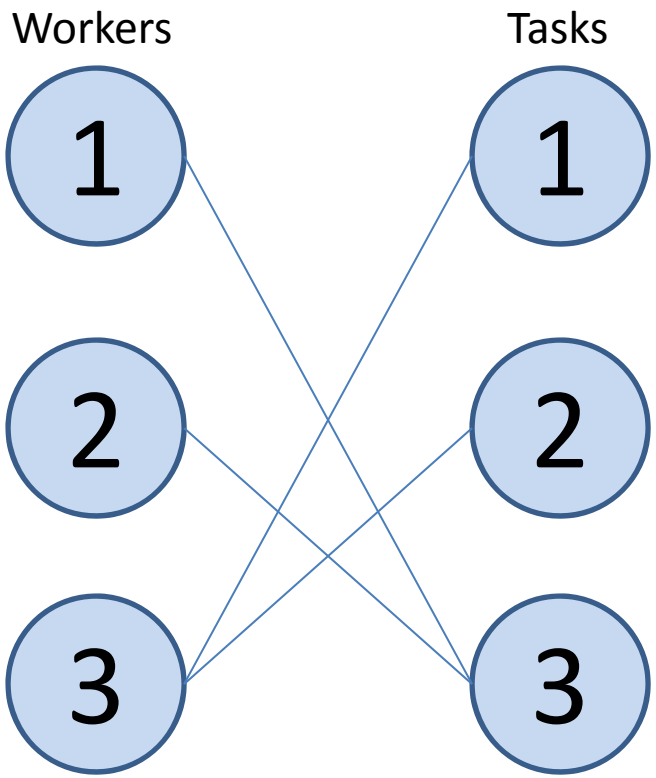


A zero-cover containing three lines:
The lines i = 1, i = 2, and j = 3

The Assignment Problem: Example 2

Another example of a zero-cover





A zero-cover containing two lines:
The line i = 3 and the line j = 3

The Assignment Problem: Towards the Hungarian Algorithm

Claim 2

Given an n x n cost table for the assignment problem,

there is no all-zero-assignment if and only if there is a zero-cover that uses fewer than n lines.

The Assignment Problem: Towards the Hungarian Algorithm

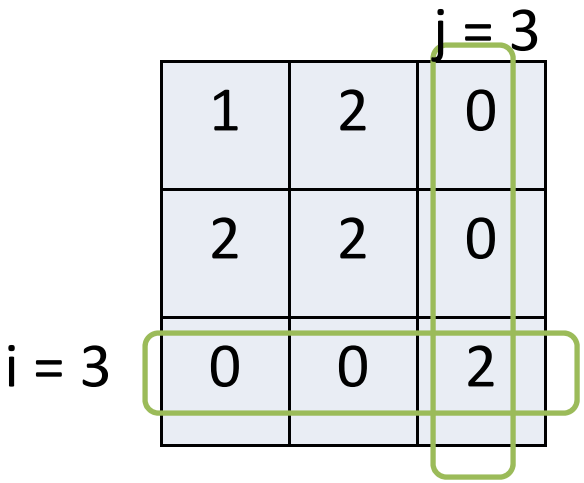
Claim 2

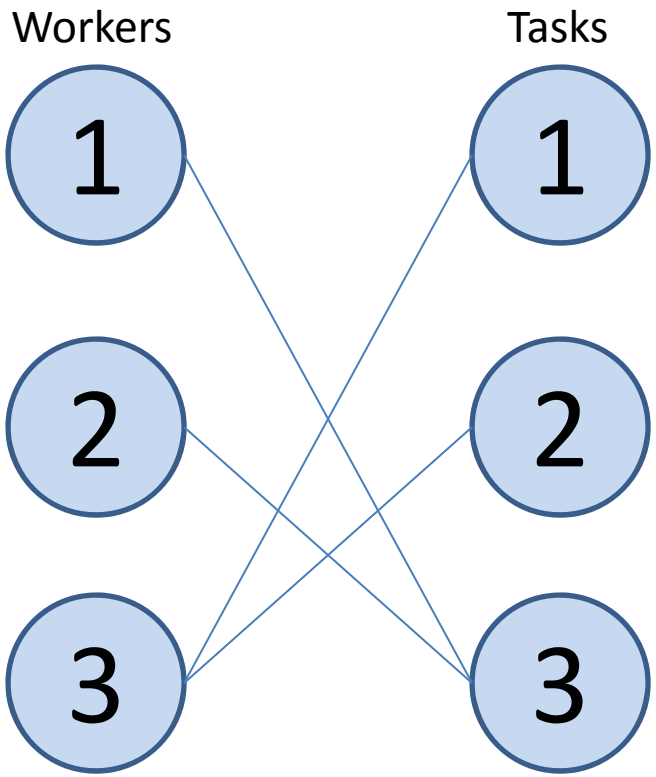
Given an n x n cost table for the assignment problem,

there is no all-zero-assignment if and only if there is a zero-cover that uses fewer than n lines.

Proof? (Homework 4!)

The Assignment Problem: Example 2





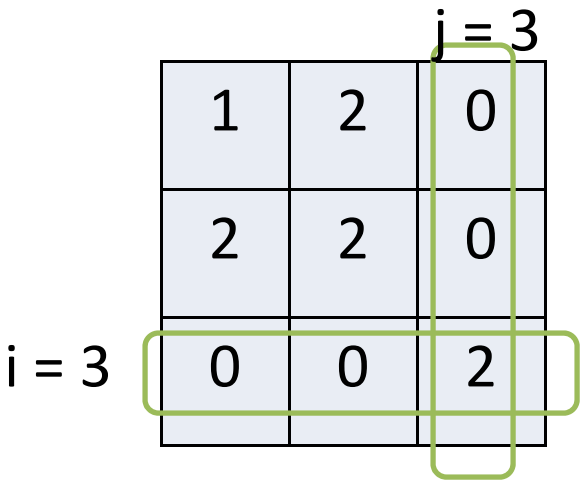
A zero-cover containing two lines:
The line i = 3 and the line j = 3

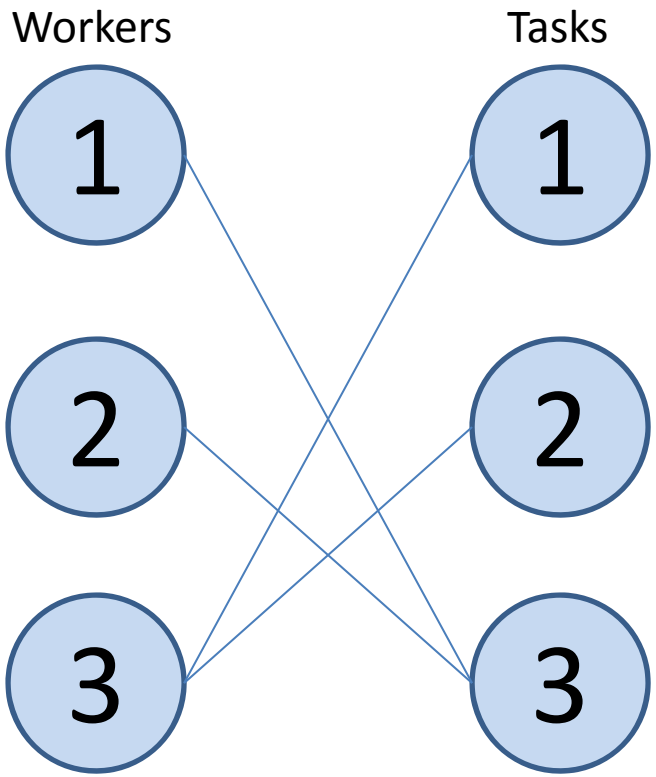
The Hungarian Algorithm

0. Set up a cost table. Assume all costs are ≥ 0.
1. For each row i = 1, …, n
   let α_i = the smallest entry in row i
   update: $t_{ij} \leftarrow t_{ij} - \alpha_i$ for all entries (i,j) in row i
2. For each col j = 1, …, n
   let β_j = the smallest entry in column j
   update: $t_{ij} \leftarrow t_{ij} - \beta_j$ for all entries (i,j) in col j
3. Check if there is an all-zero assignment
   If there is one, this assignment is optimal
   If there isn't one:
   - find a zero-cover containing fewer than n lines,
   - let δ = the smallest entry not covered by any lines
   - update:
     $t_{ij} \leftarrow t_{ij} - \delta$ if entry (i, j) is not covered by any lines
     $t_{ij} \leftarrow t_{ij}$ if entry (i, j) is covered by exactly one line
     $t_{ij} \leftarrow t_{ij} + \delta$ if entry (i, j) is covered by exactly two lines

   (Repeat step 3 until there is an all-zero assignment)

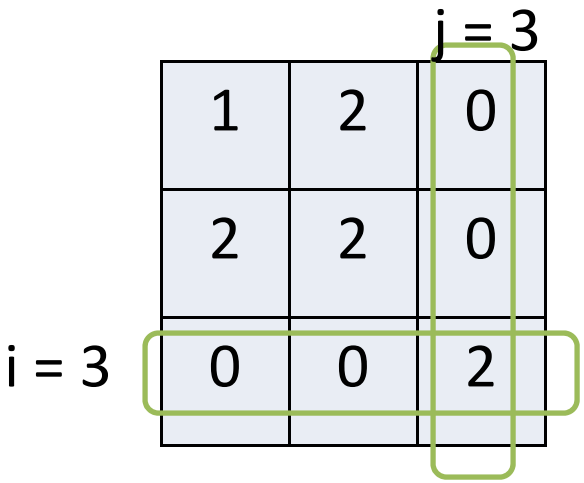
The Assignment Problem: Example 2

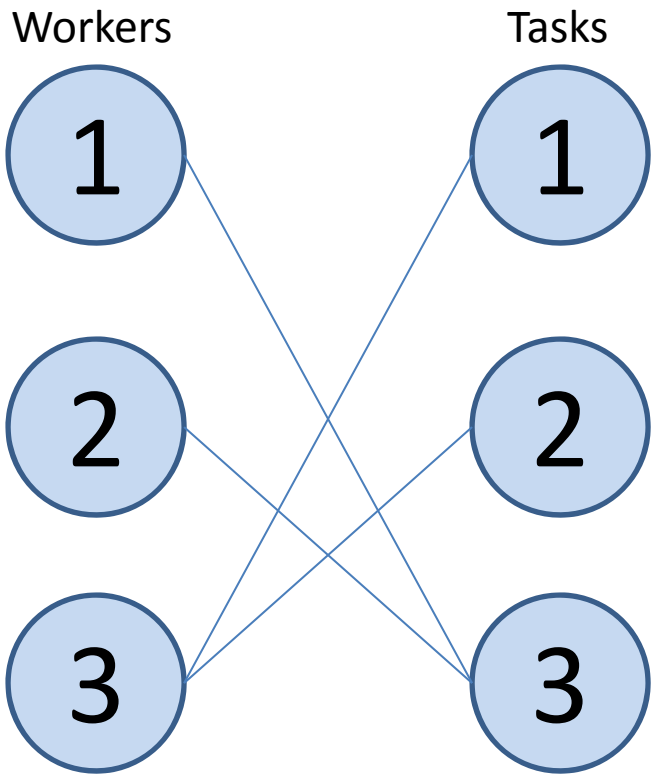




A zero-cover containing two lines:
The line i = 3 and the line j = 3

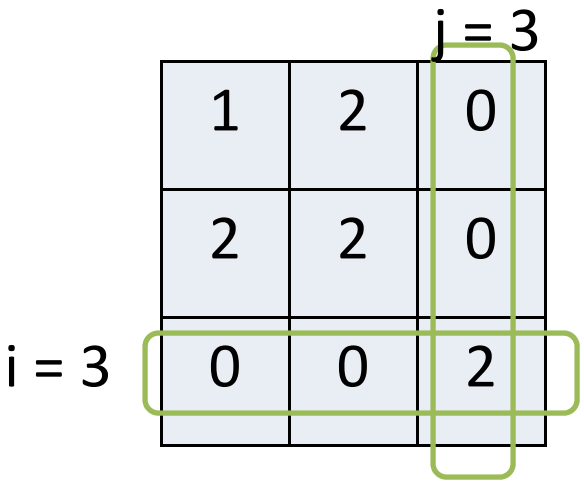
The Assignment Problem: Example 2

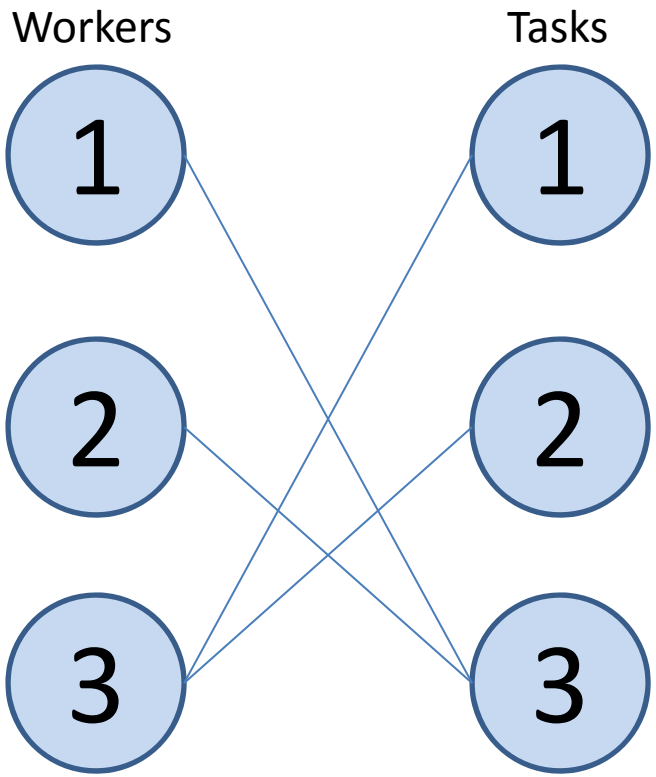




δ = 1 = smallest number not covered by any line

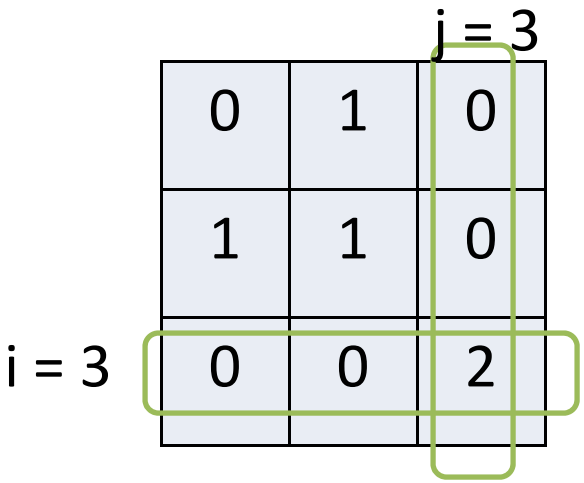
The Assignment Problem: Example 2

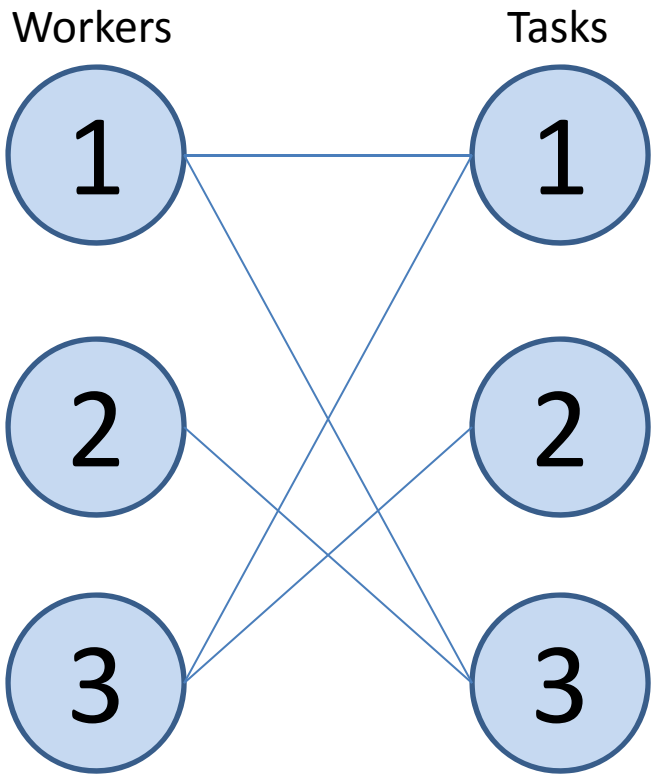




$t_{ij} \leftarrow t_{ij} - \delta$	**if entry (i, j) is not covered by any lines**
$t_{ij} \leftarrow t_{ij}$	if entry (i, j) is covered by exactly one line
$t_{ij} \leftarrow t_{ij} + \delta$	if entry (i, j) is covered by exactly two lines

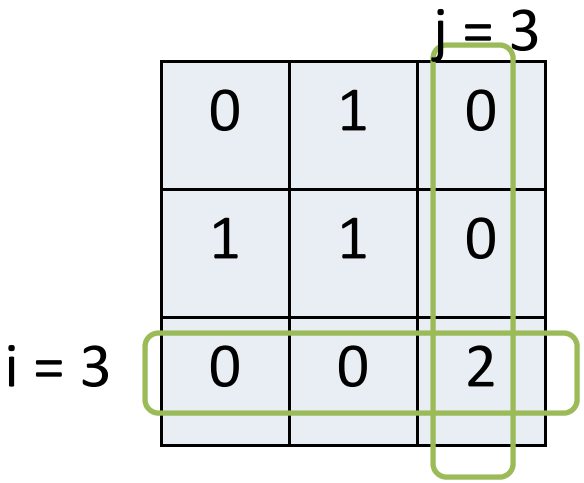
The Assignment Problem: Example 2

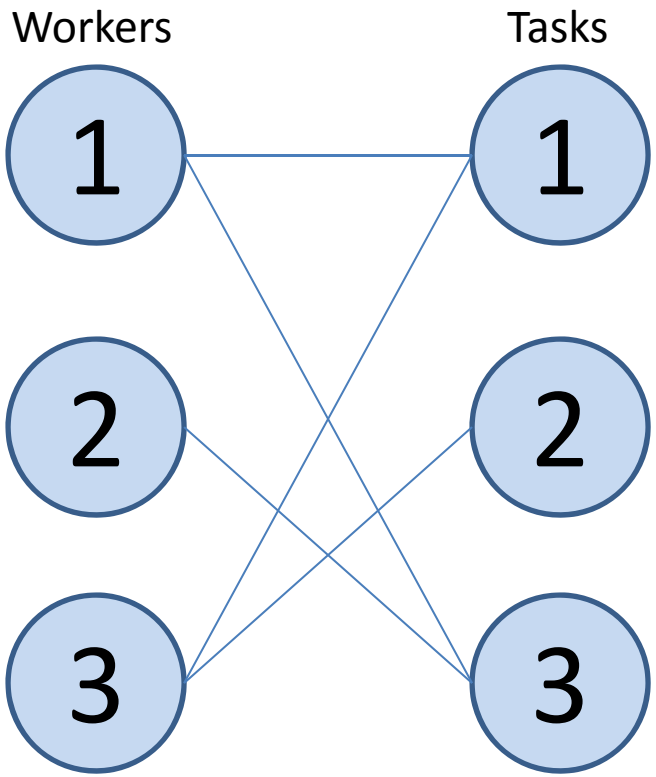




$t_{ij} \leftarrow t_{ij} - \delta$	**if entry (i, j) is not covered by any lines**
$t_{ij} \leftarrow t_{ij}$	if entry (i, j) is covered by exactly one line
$t_{ij} \leftarrow t_{ij} + \delta$	if entry (i, j) is covered by exactly two lines

The Assignment Problem: Example 2



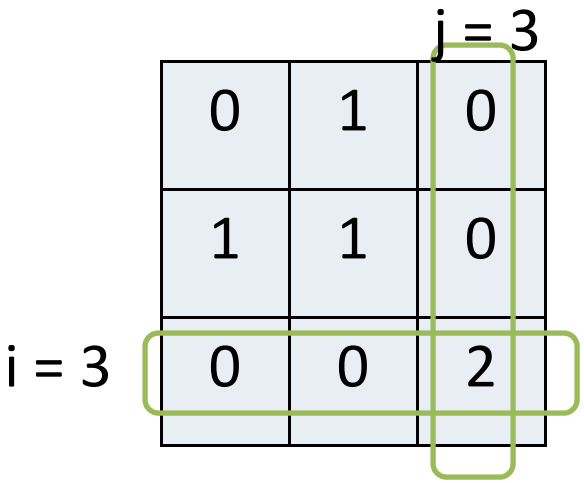


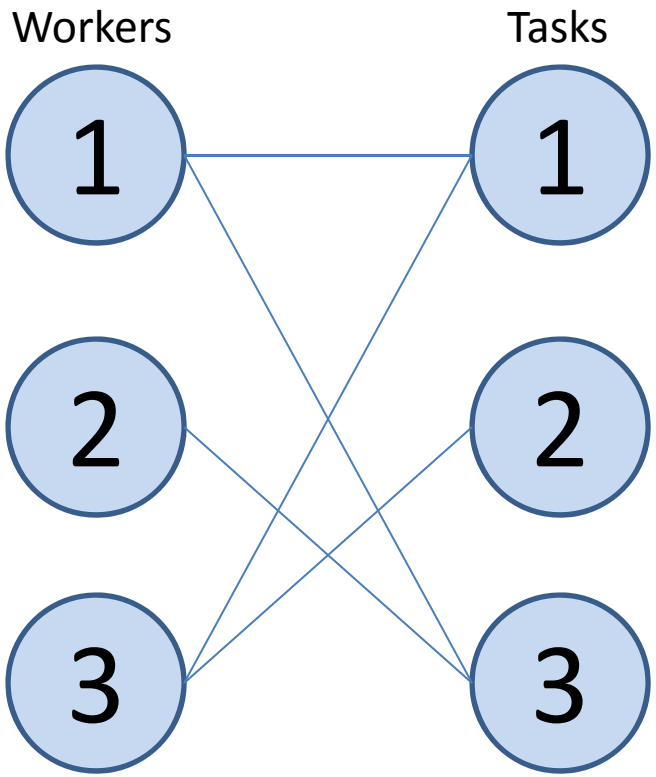
$t_{ij} \leftarrow t_{ij} - \delta$ if entry (i, j) is not covered by any lines

$t_{ij} \leftarrow t_{ij}$ if entry (i, j) is covered by exactly one line

$t_{ij} \leftarrow t_{ij} + \delta$ if entry (i, j) is covered by exactly two lines

The Assignment Problem: Example 2

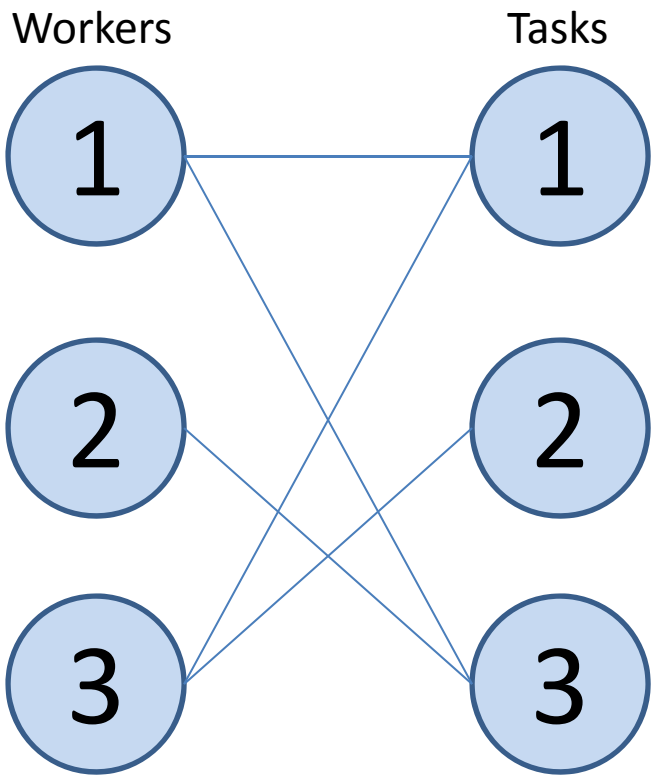




$t_{ij} \leftarrow t_{ij} - \delta$	if entry (i, j) is not covered by any lines
$t_{ij} \leftarrow t_{ij}$	if entry (i, j) is covered by exactly one line
$t_{ij} \leftarrow t_{ij} + \delta$	**if entry (i, j) is covered by exactly two lines**

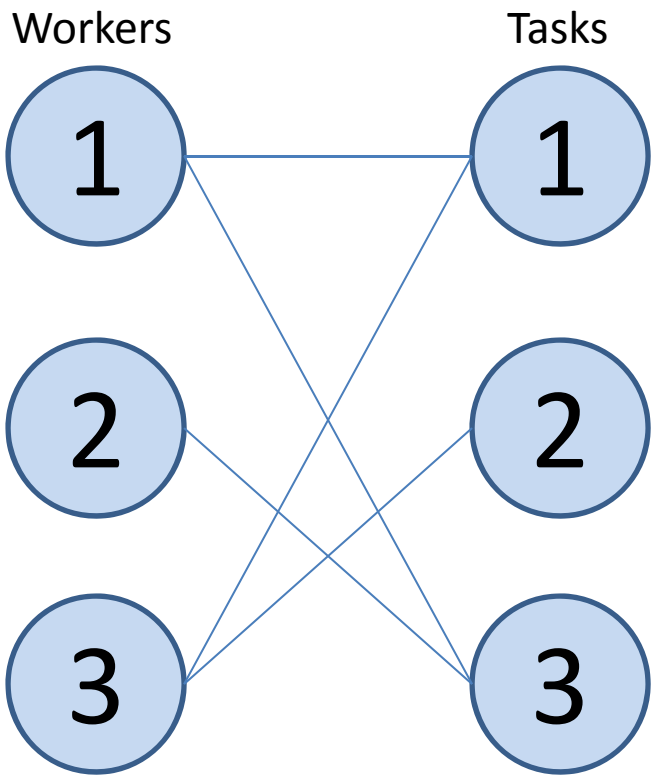
The Assignment Problem: Example 2

0	1	0
1	1	0
0	0	3



The Assignment Problem: Example 2

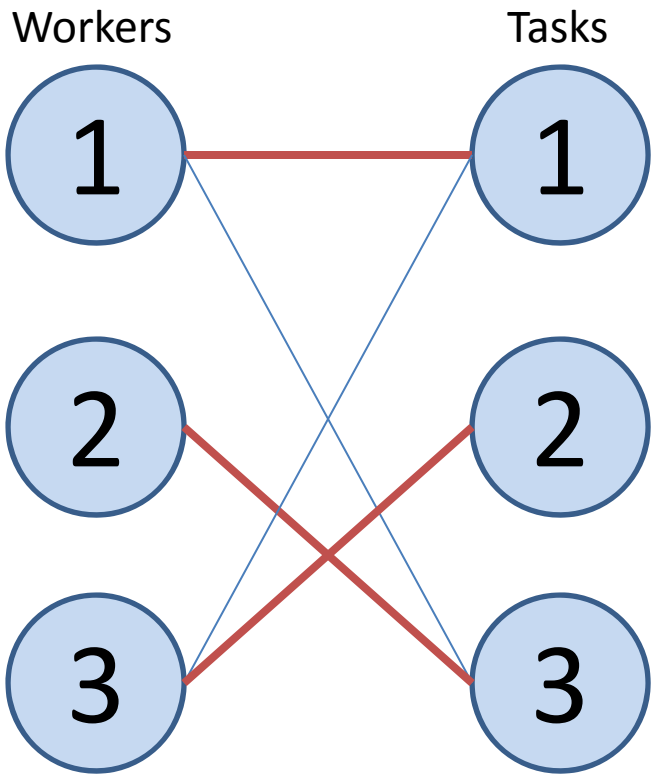
0	1	0
1	1	0
0	0	3



Find an all-zero-assignment?

The Assignment Problem: Example 2

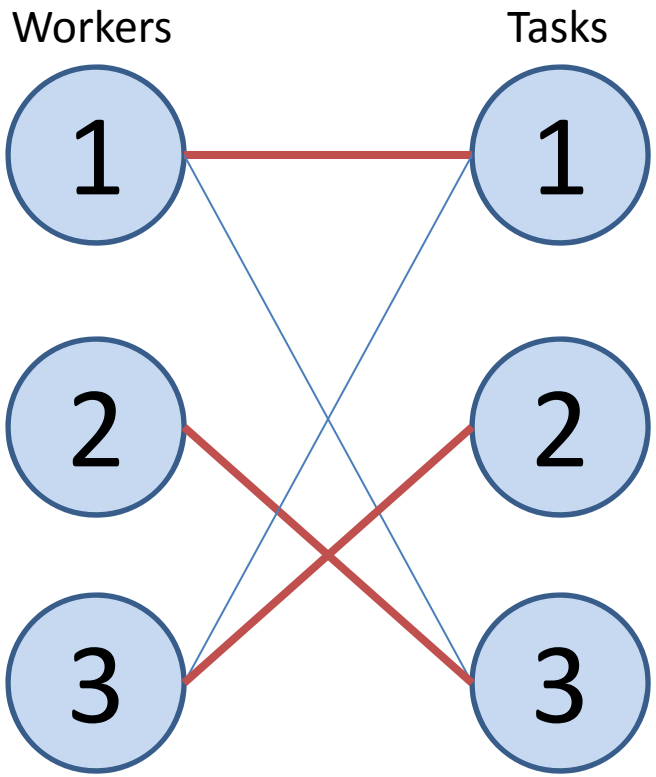
0	1	0
1	1	0
0	0	3



An all-zero assignment: (1, 1), (2, 3), (3, 2)

The Assignment Problem: Example 2

8	10	7
10	11	8
5	6	7



The assignment: (1, 1), (2, 3), (3, 2)
is also optimal for the original problem,
with total cost = 8 + 8 + 6 = 22

i>clicker

(after the break)

Q1:

The Hungarian Algorithm

0. Set up a cost table. Assume all costs are ≥ 0.
1. For each row i = 1, …, n
   let α_i = the smallest entry in row i
   update: $t_{ij} \leftarrow t_{ij} - \alpha_i$ for all entries (i,j) in row i
2. For each col j = 1, …, n
   let β_j = the smallest entry in column j
   update: $t_{ij} \leftarrow t_{ij} - \beta_j$ for all entries (i,j) in col j
3. Check if there is an all-zero assignment
   If there is one, this assignment is optimal
   If there isn't one:
   - find a zero-cover containing fewer than n lines,
   - let δ = the smallest entry not covered by any lines
   - update:
     $t_{ij} \leftarrow t_{ij} - \delta$ if entry (i, j) is not covered by any lines
     $t_{ij} \leftarrow t_{ij}$ if entry (i, j) is covered by exactly one line
     $t_{ij} \leftarrow t_{ij} + \delta$ if entry (i, j) is covered by exactly two lines

   (Repeat step 3 until there is an all-zero assignment)

The Hungarian Algorithm

Loose ends:

- After updating the cost table,
  - how do we check if there is an all-zero-assignment?
  - Is this an “easy” step?

The Hungarian Algorithm

Claim 2

Given an n x n cost table for the assignment problem,

there is no all-zero-assignment if and only if there is a zero-cover that uses fewer than n lines.

Proof? (Homework 4!)

The Hungarian Algorithm

Loose ends:

- Does the algorithm terminate eventually?
- Does the algorithm produce an optimal assignment when it terminates?
- How many iterations until it terminates?
- On the assumption that $t_{ij} \geq 0$:
  How if there are negative costs
  (if there are pairs with $t_{ij} < 0$)?

The Hungarian Algorithm

Loose ends:

- Does the algorithm terminate eventually? Yes!
- Does the algorithm produce an optimal assignment when it terminates? Yes!
- How many iterations until we stop? Not too many!
- On the assumption that $t_{ij} \geq 0$:
  How if there are negative costs
  (if there are pairs with $t_{ij} < 0$)? Not a problem!

Wrapping up Network Optimization

- Models/Problems:
  - Minimum-cost network flow (“mincost flow”)
  - Maximum flow (“maxflow”)
  - The shortest path problem
  - The assignment problem
  - The longest-path problem
  - Maximum bipartite matching
  - Bipartite minimum vertex-cover
  - Lots more!

Wrapping up Network Optimization

- Algorithms:
  - The Ford-Fulkerson Algorithm
  - The Hungarian Algorithm
  - Simplex method (when network problems are written as linear programs)

Wrapping up Network Optimization

- Theorems and other results
  - The integrality theorem for mincost flow
  - The integrality theorem for maxflow
  - The maxflow/mincut theorem
  - …

Wrapping up Network Optimization

- Other new stuff in your "Optimizator's Purse":
  - Modeling and solving using AMPL
  - How to formulate a problem as a given model (the 3-step framework!)
    - Formulating network optimization problems as LPs
    - Formulating various problems as mincost flow
    - Formulating various problems as maxflow
  - Utilizing LP duality to analyze network problems

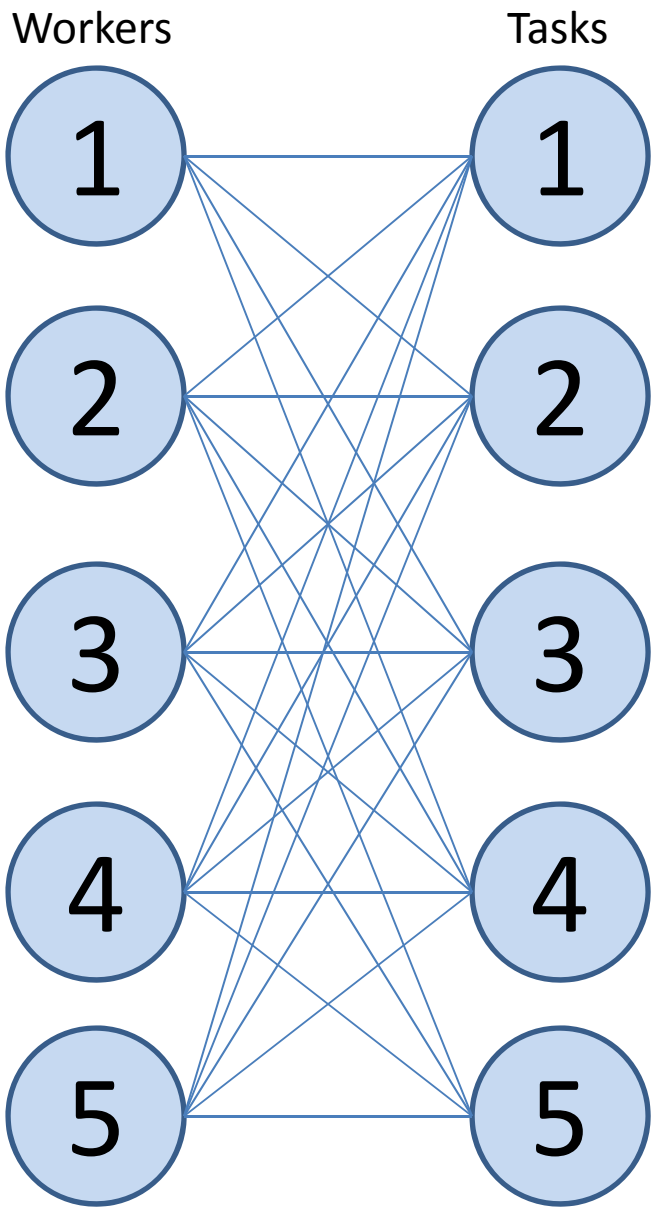
Prelim 1: Thursday April 21, 7:30pm – 9pm (1.5 hrs), Kimball B11

- All network optimization topics, including the assignment problem and the Hungarian algorithm.
- HW 4 will include these topics. Solutions will be posted on Monday after the HW is due.
- Practice prelim will be posted.
- Recitations next week will consist of review and overview of practice prelim solutions.
- Types of questions will have similar flavors as homework problems.
- Check for prelim conflicts and let me know ASAP.

Q2: not i>clicker

The Assignment Problem: Example 3

3	5	6	2	2
2	3	5	3	2
3	0	4	2	2
3	0	3	2	2
0	3	0	1	0



The Hungarian Algorithm

0. Set up a cost table. Assume all costs are ≥ 0.
1. For each row i = 1, …, n
   let α_i = the smallest entry in row i
   update: $t_{ij} \leftarrow t_{ij} - \alpha_i$ for all entries (i,j) in row i
2. For each col j = 1, …, n
   let β_j = the smallest entry in column j
   update: $t_{ij} \leftarrow t_{ij} - \beta_j$ for all entries (i,j) in col j
3. Check if there is an all-zero assignment
   If there is one, this assignment is optimal
   If there isn't one:
   - find a zero-cover containing fewer than n lines,
   - let δ = the smallest entry not covered by any lines
   - update:
     $t_{ij} \leftarrow t_{ij} - \delta$ if entry (i, j) is not covered by any lines
     $t_{ij} \leftarrow t_{ij}$ if entry (i, j) is covered by exactly one line
     $t_{ij} \leftarrow t_{ij} + \delta$ if entry (i, j) is covered by exactly two lines

   (Repeat step 3 until there is an all-zero assignment)

Next time in Opt 2:
On to Dynamic Programming!