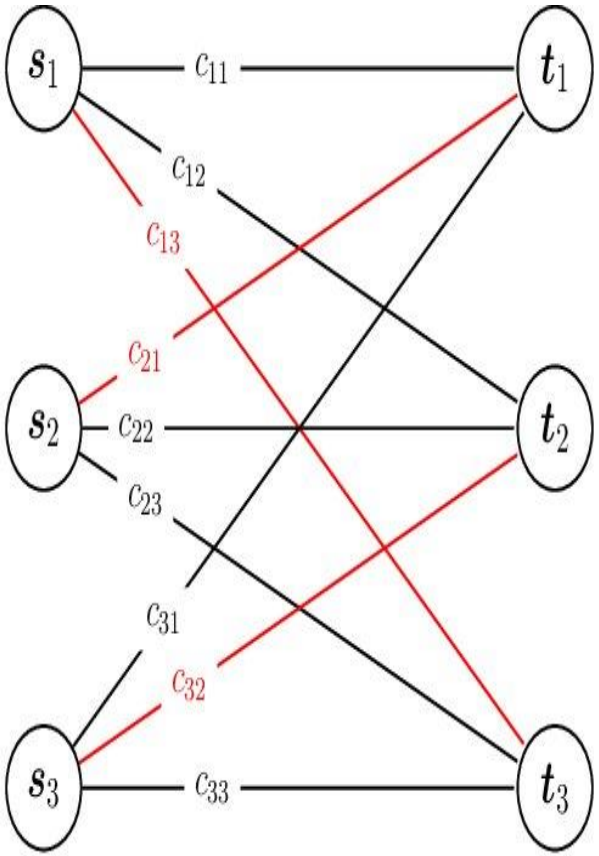
Part V: Assignment Problem

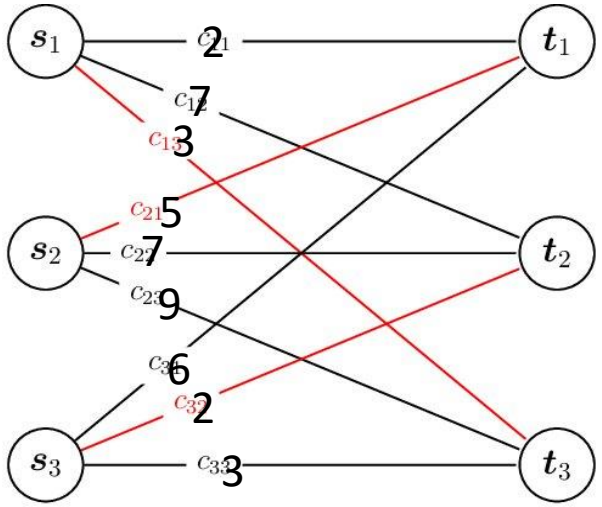
Introduction

- The **assignment problem** (in Operations Research and Combinatorial Optimization) consists of **optimally** assigning tasks to agents.
- Each agent can perform exactly one task for a given cost, and each task must be carried out by exactly one agent.
- Each assignment (agent-task pair) has a defined cost.
- The goal is to **minimize** the total cost of the assignments in order to complete all the tasks.



introduction

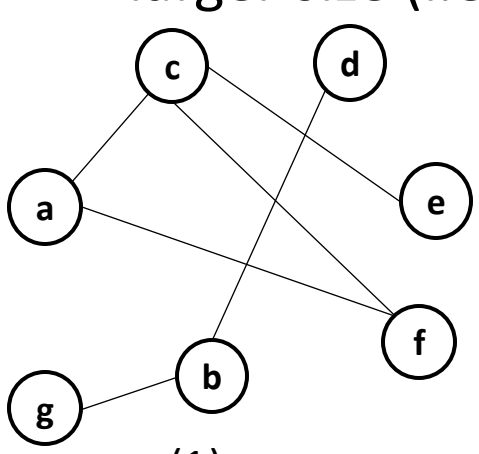
- More formally, the objective is to determine a **matching** of a size equal to the number of tasks, with minimum total weight in a weighted bipartite graph.
- If there are as many agents as tasks, it involves determining a **perfect matching** of minimum total weight in a weighted bipartite graph.
- **Exemples:**
  - Job Assignment in a Factory
  - Assigning Drivers to Delivery Routes
  - Bandwidth Allocation to Network Flows
  - Assignment of IP Addresses



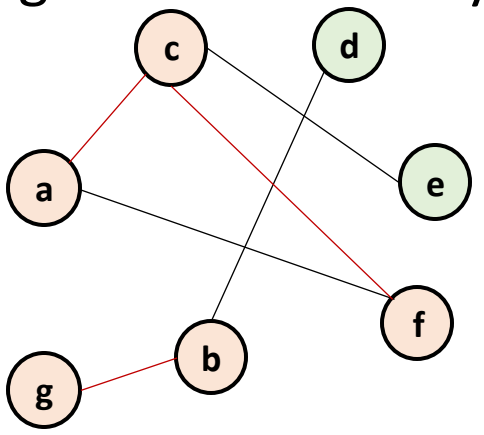
Min ∑ ci = 3+5+2 = 8

Problem Formalization

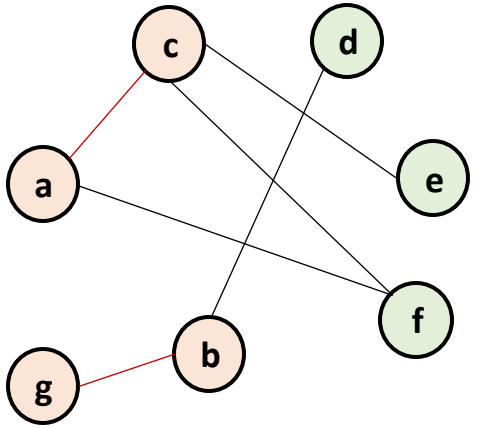
- Let G(V,E) be a simple graph. A **matching** is a set of edges such that no two edges in the set share a common vertex.
- A vertex is said to be **saturated** by a matching if it is an endpoint of an edge in that matching.
- A matching is called **perfect** if it saturates all the vertices of the graph.
- A matching is said to be **maximum** if it is impossible to find a matching with a larger size (i.e. with greater cardinality).



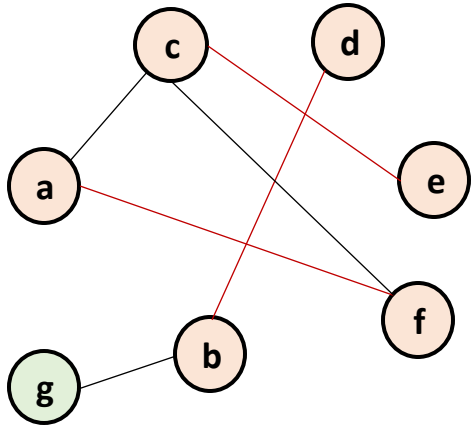
(1)



(2)



(3)

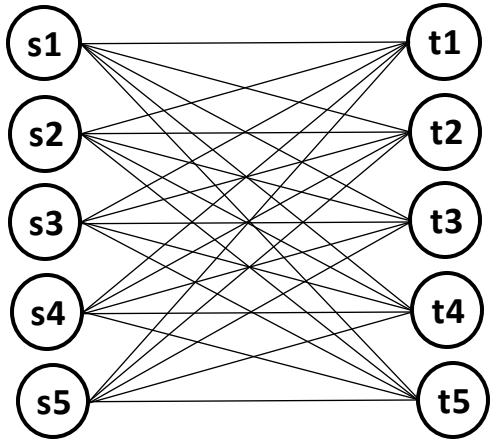


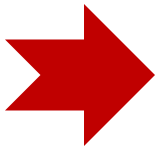
(4)

Problem Formalization

- Generalization of the **maximum matching problem** in a bipartite graph, as it involves finding, among the matchings of maximum cardinality, the one with the **minimum (or maximum) total weight**.
- Given a set of agents **S** and a set of tasks **T**, the problem is modeled by a **bipartite graph** G((S,T),E), with a weight function on the edges $c:E \rightarrow R$. The **assignment problem** then consists of finding a matching $F \subseteq E$ with $|F|=|T|$ that minimizes the sum $\sum c(e)$ of the weights of the edges in F.
- The problem can also be represented by an **assignment matrix** C=(cij).

Problem Formalization





	t1	t2	t3	t4	t5
s1	2	2	11	9	9
s2	2	3	9	10	3
s3	4	10	5	3	6
s4	2	7	5	3	6
s5	5	1	9	2	10

Problem solution

- The algorithm is applied to the assignment matrix **A**, which is a square matrix, so it may be necessary to add dummy rows or columns if needed.
- **A[i, j]** represents the cost or gain of assigning agent **i** to task **j**.
- The **objective** is to select one element per row and per column in such a way that the sum of the associated values is optimal.

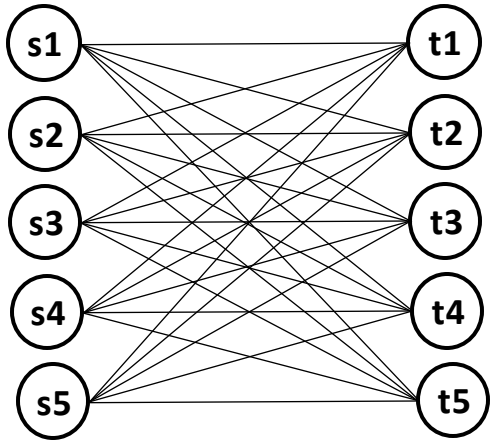
Hangarian algorithm

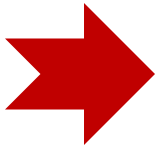
Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

Step 2: Preparation for modifying the matrix

Step 3: Modification of the matrix and return to **Step 1**





	t1	t2	t3	t4	t5
s1	2	2	11	9	9
s2	2	3	9	10	3
s3	4	10	5	3	6
s4	2	7	5	3	6
s5	5	1	9	2	10

Hangarian algorithm

Step 0: Reduction of the initial matrix

a) Find the min value of each row and subtract it from all the elements in that row

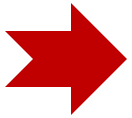
b) Find the min value of each column and subtract it from all the elements in that column

Step 1: Optimal assignment, or move to **Step 2** if impossible

Step 2: Preparation for modifying the matrix

Step 3: Modification of the matrix and return to **Step 1**

	t1	t2	t3	t4	t5	min
s1	2	2	11	9	9	**2**
s2	2	3	9	10	3	**2**
s3	4	10	5	3	6	**3**
s4	2	7	5	3	6	**2**
s5	5	1	9	2	10	**1**



	t1	t2	t3	t4	t5
s1	0	0	9	7	7
s2	0	1	7	8	1
s3	1	7	2	0	3
s4	0	5	3	1	4
s5	4	0	8	1	9

Hangarian algorithm

Step 0: Reduction of the initial matrix

a) Find the min value of each row and subtract it from all the elements in that row

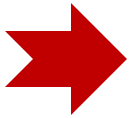
b) Find the min value of each column and subtract it from all the elements in that column

Step 1: Optimal assignment, or move to **Step 2** if impossible

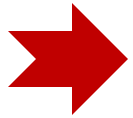
Step 2: Preparation for modifying the matrix

Step 3: Modification of the matrix and return to **Step 1**

	t1	t2	t3	t4	t5	min
s1	2	2	11	9	9	**2**
s2	2	3	9	10	3	**2**
s3	4	10	5	3	6	**3**
s4	2	7	5	3	6	**2**
s5	5	1	9	2	10	**1**



	t1	t2	t3	t4	t5
s1	0	0	9	7	7
s2	0	1	7	8	1
s3	1	7	2	0	3
s4	0	5	3	1	4
s5	4	0	8	1	9
min	**0**	**0**	**2**	**0**	**1**



	t1	t2	t3	t4	t5
s1	0	0	7	7	6
s2	0	1	5	8	0
s3	1	7	0	0	2
s4	0	5	1	1	3
s5	4	0	6	1	8

Hangarian algorithm

Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

a) Find the row containing the fewest uncrossed zeros (arbitrarily the highest one if tied)
b) Circle one of its zeros (arbitrarily the leftmost one)
c) Cross out all other zeros in the same row and column as the circled zero
d) Repeat the same steps until there are no more zeros to circle or cross out
e) If there's one zero per row and per column, the assignment is optimal; otherwise, proceed to Step 2

Step 2: Preparation for modifying the matrix

Step 3: Modification of the matrix and return to **Step 1**

	t1	t2	t3	t4	t5
s1	~~0~~	0	7	7	6
s2	~~0~~	1	5	8	0
s3	1	7	0	0	2
s4	0	5	1	1	3
s5	4	0	6	1	8

Hangarian algorithm

Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

a) Find the row containing the fewest uncrossed zeros (arbitrarily the highest one if tied)
b) Circle one of its zeros (arbitrarily the leftmost one)
c) Cross out all other zeros in the same row and column as the circled zero
d) Repeat the same steps until there are no more zeros to circle or cross out
e) If there's one zero per row and per column, the assignment is optimal; otherwise, proceed to Step 2

Step 2: Preparation for modifying the matrix

Step 3: Modification of the matrix and return to **Step 1**

	t1	t2	t3	t4	t5
s1	~~0~~	[0]	7	7	6
s2	~~0~~	1	5	8	0
s3	1	7	0	0	2
s4	[0]	5	1	1	3
s5	4	~~0~~	6	1	8

Hangarian algorithm

Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

a) Find the row containing the fewest uncrossed zeros (arbitrarily the highest one if tied)
b) Circle one of its zeros (arbitrarily the leftmost one)
c) Cross out all other zeros in the same row and column as the circled zero
d) Repeat the same steps until there are no more zeros to circle or cross out
e) If there's one zero per row and per column, the assignment is optimal; otherwise, proceed to Step 2

Step 2: Preparation for modifying the matrix

Step 3: Modification of the matrix and return to **Step 1**

	t1	t2	t3	t4	t5
s1	~~0~~	[0]	7	7	6
s2	~~0~~	1	5	8	[0]
s3	1	7	0	0	2
s4	[0]	5	1	1	3
s5	4	~~0~~	6	1	8

Hangarian algorithm

Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

a) Find the row containing the fewest uncrossed zeros (arbitrarily the highest one if tied)
b) Circle one of its zeros (arbitrarily the leftmost one)
c) Cross out all other zeros in the same row and column as the circled zero
d) Repeat the same steps until there are no more zeros to circle or cross out
e) If there's one zero per row and per column, the assignment is optimal; otherwise, proceed to Step 2

Step 2: Preparation for modifying the matrix

Step 3: Modification of the matrix and return to **Step 1**

	t1	t2	t3	t4	t5
s1	~~0~~	[0]	7	7	6
s2	~~0~~	1	5	8	[0]
s3	1	7	[0]	~~0~~	2
s4	[0]	5	1	1	3
s5	4	~~0~~	6	1	8

The assignment is not optimal !

Hangarian algorithm

Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

Step 2: Preparation for modifying the matrix

a) Mark every row that does not contain a circled zero
b) Mark every column that has a crossed zero in a marked row
c) Mark every row that has a circled zero in a marked column
d) Repeat steps (b) and (c) until no more markings are possible
e) Draw a line through every unmarked row and every marked column

Step 3: Modification of the matrix and return to **Step 1**

	t1	t2	t3	t4	t5	
s1	~~0~~	[0]	7	7	6	
s2	~~0~~	1	5	8	[0]	
s3	1	7	[0]	~~0~~	2	
s4	[0]	5	1	1	3	
s5	4	~~0~~	6	1	8	X

Hangarian algorithm

Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

Step 2: Preparation for modifying the matrix

a) Mark every row that does not contain a circled zero
b) Mark every column that has a crossed zero in a marked row
c) Mark every row that has a circled zero in a marked column
d) Repeat steps (b) and (c) until no more markings are possible
e) Draw a line through every unmarked row and every marked column

Step 3: Modification of the matrix and return to **Step 1**

X

	t1	t2	t3	t4	t5	
s1	~~0~~	[0]	7	7	6	
s2	~~0~~	1	5	8	[0]	
s3	1	7	[0]	~~0~~	2	
s4	[0]	5	1	1	3	
s5	4	~~0~~	6	1	8	X

Hangarian algorithm

Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

Step 2: Preparation for modifying the matrix

a) Mark every row that does not contain a circled zero
b) Mark every column that has a crossed zero in a marked row
c) Mark every row that has a circled zero in a marked column
d) Repeat steps (b) and (c) until no more markings are possible
e) Draw a line through every unmarked row and every marked column

Step 3: Modification of the matrix and return to **Step 1**

		X			
	t1	**t2**	**t3**	**t4**	**t5**
s1	~~0~~	[0]	7	7	6	X
s2	~~0~~	1	5	8	[0]	
s3	1	7	[0]	~~0~~	2	
s4	[0]	5	1	1	3	
s5	4	~~0~~	6	1	8	X

Hangarian algorithm

Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

Step 2: Preparation for modifying the matrix

a) Mark every row that does not contain a circled zero
b) Mark every column that has a crossed zero in a marked row
c) Mark every row that has a circled zero in a marked column
d) Repeat steps (b) and (c) until no more markings are possible
e) Draw a line through every unmarked row and every marked column

Step 3: Modification of the matrix and return to **Step 1**

		X	X			
	t1	**t2**	**t3**	**t4**	**t5**	
s1	~~0~~	[0]	7	7	6	X
s2	~~0~~	1	5	8	[0]	
s3	1	7	[0]	~~0~~	2	
s4	[0]	5	1	1	3	
s5	4	~~0~~	6	1	8	X

Hangarian algorithm

Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

Step 2: Preparation for modifying the matrix

a) Mark every row that does not contain a circled zero
b) Mark every column that has a crossed zero in a marked row
c) Mark every row that has a circled zero in a marked column
d) Repeat steps (b) and (c) until no more markings are possible
e) Draw a line through every unmarked row and every marked column

Step 3: Modification of the matrix and return to **Step 1**

		X	X			
	t1	**t2**	**t3**	**t4**	**t5**	
s1	~~0~~	[0]	7	7	6	X
s2	~~0~~	1	5	8	[0]	
s3	1	7	[0]	~~0~~	2	
s4	[0]	5	1	1	3	X
s5	4	~~0~~	6	1	8	X

Hangarian algorithm

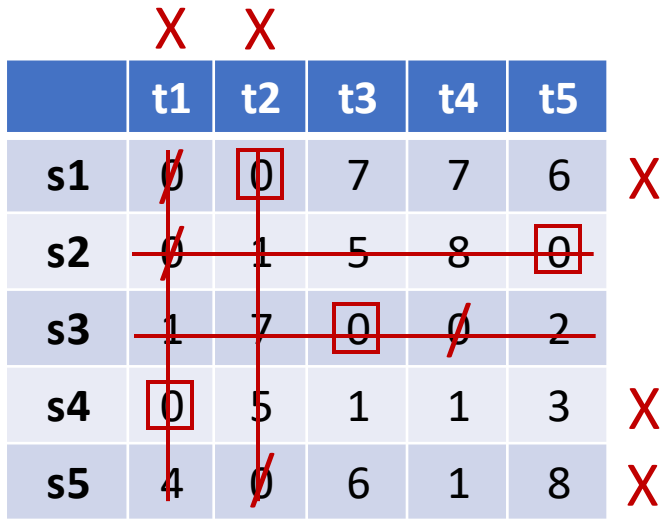
Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

Step 2: Preparation for modifying the matrix

a) Mark every row that does not contain a circled zero
b) Mark every column that has a crossed zero in a marked row
c) Mark every row that has a circled zero in a marked column
d) Repeat steps (b) and (c) until no more markings are possible
e) Draw a line through every unmarked row and every marked column

Step 3: Modification of the matrix and return to **Step 1**



Hangarian algorithm

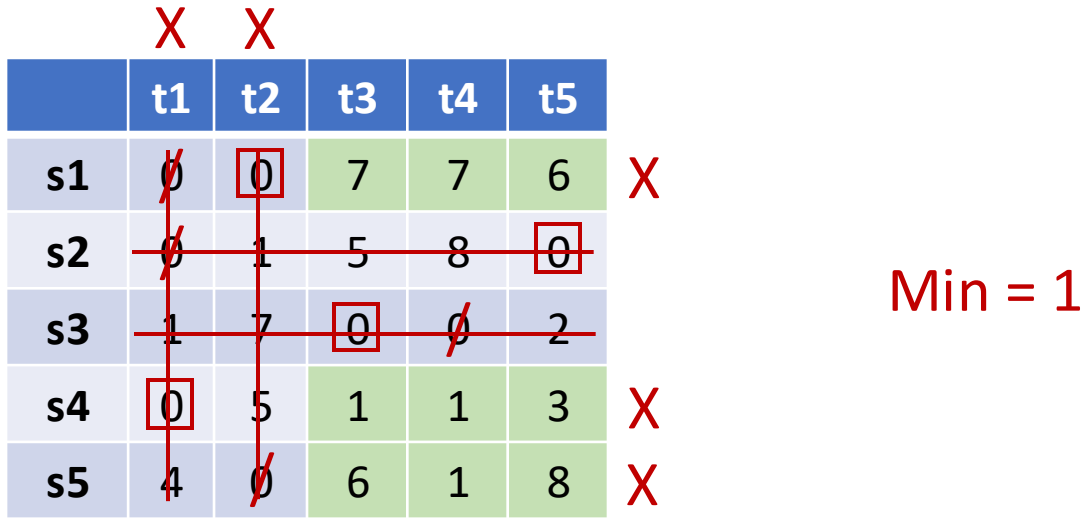
Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

Step 2: Preparation for modifying the matrix

Step 3: Modification of the matrix and return to **Step 1**

a) Find the min value among the unlined (uncovered) cells
b) Subtract this value from all unlined cells and add it to all doubly lined cells
c) Go back to Step 1



Hangarian algorithm

Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

Step 2: Preparation for modifying the matrix

Step 3: Modification of the matrix and return to **Step 1**

a) Find the min value among the unlined (uncovered) cells
b) Subtract this value from all unlined cells and add it to all doubly lined cells
c) Go back to Step 1

	t1	t2	t3	t4	t5
s1	0	0	6	6	5
s2	1	2	5	8	0
s3	2	8	0	0	2
s4	0	5	0	0	2
s5	4	0	5	0	7

Min = 1

Hangarian algorithm

Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

a) Find the row containing the fewest uncrossed zeros (arbitrarily the highest one if tied)
b) Circle one of its zeros (arbitrarily the leftmost one)
c) Cross out all other zeros in the same row and column as the circled zero
d) Repeat the same steps until there are no more zeros to circle or cross out
e) If there's one zero per row and per column, the assignment is optimal; otherwise, proceed to Step 2

Step 2: Preparation for modifying the matrix

Step 3: Modification of the matrix and return to **Step 1**

	t1	t2	t3	t4	t5
s1	0	0	6	6	5
s2	1	2	5	8	0
s3	2	8	0	0	2
s4	0	5	0	0	2
s5	4	0	5	0	7

Hangarian algorithm

Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

a) Find the row containing the fewest uncrossed zeros (arbitrarily the highest one if tied)
b) Circle one of its zeros (arbitrarily the leftmost one)
c) Cross out all other zeros in the same row and column as the circled zero
d) Repeat the same steps until there are no more zeros to circle or cross out
e) If there's one zero per row and per column, the assignment is optimal; otherwise, proceed to Step 2

Step 2: Preparation for modifying the matrix

Step 3: Modification of the matrix and return to **Step 1**

	t1	t2	t3	t4	t5
s1	0	~~0~~	6	6	5
s2	1	2	5	8	0
s3	2	8	0	0	2
s4	~~0~~	5	0	0	2
s5	4	0	5	0	7

Hangarian algorithm

Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

a) Find the row containing the fewest uncrossed zeros (arbitrarily the highest one if tied)
b) Circle one of its zeros (arbitrarily the leftmost one)
c) Cross out all other zeros in the same row and column as the circled zero
d) Repeat the same steps until there are no more zeros to circle or cross out
e) If there's one zero per row and per column, the assignment is optimal; otherwise, proceed to Step 2

Step 2: Preparation for modifying the matrix

Step 3: Modification of the matrix and return to **Step 1**

	t1	t2	t3	t4	t5
s1	[0]	~~0~~	6	6	5
s2	1	2	5	8	[0]
s3	2	8	[0]	~~0~~	2
s4	~~0~~	5	~~0~~	0	2
s5	4	0	5	0	7

Hangarian algorithm

Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

a) Find the row containing the fewest uncrossed zeros (arbitrarily the highest one if tied)
b) Circle one of its zeros (arbitrarily the leftmost one)
c) Cross out all other zeros in the same row and column as the circled zero
d) Repeat the same steps until there are no more zeros to circle or cross out
e) If there's one zero per row and per column, the assignment is optimal; otherwise, proceed to Step 2

Step 2: Preparation for modifying the matrix

Step 3: Modification of the matrix and return to **Step 1**

	t1	t2	t3	t4	t5
s1	[0]	~~0~~	6	6	5
s2	1	2	5	8	[0]
s3	2	8	[0]	~~0~~	2
s4	~~0~~	5	~~0~~	[0]	2
s5	4	0	5	~~0~~	7

Hangarian algorithm

Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

a) Find the row containing the fewest uncrossed zeros (arbitrarily the highest one if tied)
b) Circle one of its zeros (arbitrarily the leftmost one)
c) Cross out all other zeros in the same row and column as the circled zero
d) Repeat the same steps until there are no more zeros to circle or cross out
e) If there's one zero per row and per column, the assignment is optimal; otherwise, proceed to Step 2

Step 2: Preparation for modifying the matrix

Step 3: Modification of the matrix and return to **Step 1**

	t1	t2	t3	t4	t5
s1	[0]	~~0~~	6	6	5
s2	1	2	5	8	[0]
s3	2	8	[0]	~~0~~	2
s4	~~0~~	5	~~0~~	[0]	2
s5	4	[0]	5	~~0~~	7

Hangarian algorithm

Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

a) Find the row containing the fewest uncrossed zeros (arbitrarily the highest one if tied)
b) Circle one of its zeros (arbitrarily the leftmost one)
c) Cross out all other zeros in the same row and column as the circled zero
d) Repeat the same steps until there are no more zeros to circle or cross out
e) If there's one zero per row and per column, the assignment is optimal; otherwise, proceed to Step 2

Step 2: Preparation for modifying the matrix

Step 3: Modification of the matrix and return to **Step 1**

	t1	t2	t3	t4	t5
s1	0	0	6	6	5
s2	1	2	5	8	0
s3	2	8	0	0	2
s4	0	5	0	0	2
s5	4	0	5	0	7

Hangarian algorithm

Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

a) Find the row containing the fewest uncrossed zeros (arbitrarily the highest one if tied)
b) Circle one of its zeros (arbitrarily the leftmost one)
c) Cross out all other zeros in the same row and column as the circled zero
d) Repeat the same steps until there are no more zeros to circle or cross out
e) If there's one zero per row and per column, the assignment is optimal; otherwise, proceed to Step2

Step 2: Preparation for modifying the matrix

Step 3: Modification of the matrix and return to **Step 1**

	t1	t2	t3	t4	t5
s1	2	2	11	9	9
s2	2	3	9	10	3
s3	4	10	5	3	6
s4	2	7	5	3	6
s5	5	1	9	2	10

Optimal assignment: **{(s1,t1), (s2,t5), (s3,t3), (s4,t4), (s5,t2)}**
Cost = 2+3+5+3+1 = 14

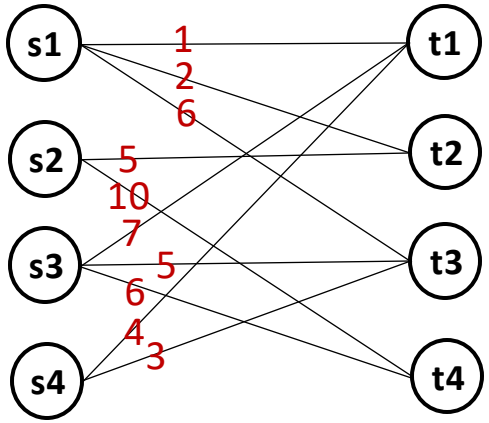
Hangarian algorithm - Examples

Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

Step 2: Preparation for modifying the matrix

Step 3: Modification of the matrix and return to **Step 1**



	t1	t2	t3	t4
s1	1	2	6	∞
s2	∞	5	∞	10
s3	7	∞	5	6
s4	4	∞	3	∞

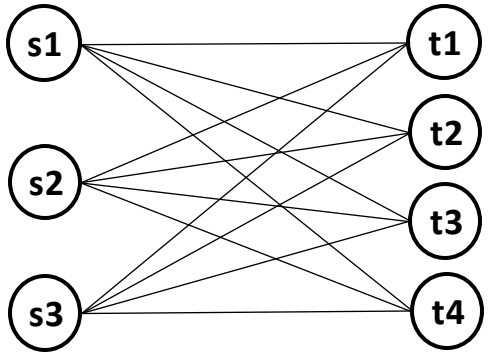
Hangarian algorithm - Examples

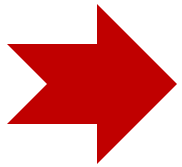
Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

Step 2: Preparation for modifying the matrix

Step 3: Modification of the matrix and return to **Step 1**





	t1	t2	t3	t4
s1	7	8	6	10
s2	5	4	13	14
s3	7	4	3	7
	0	0	0	0

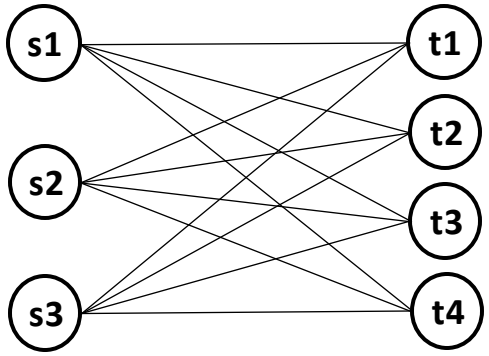
Hangarian algorithm - Examples

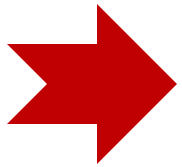
Step 0: Reduction of the initial matrix

Step 1: Optimal assignment, or move to **Step 2** if impossible

Step 2: Preparation for modifying the matrix

Step 3: Modification of the matrix and return to **Step 1**





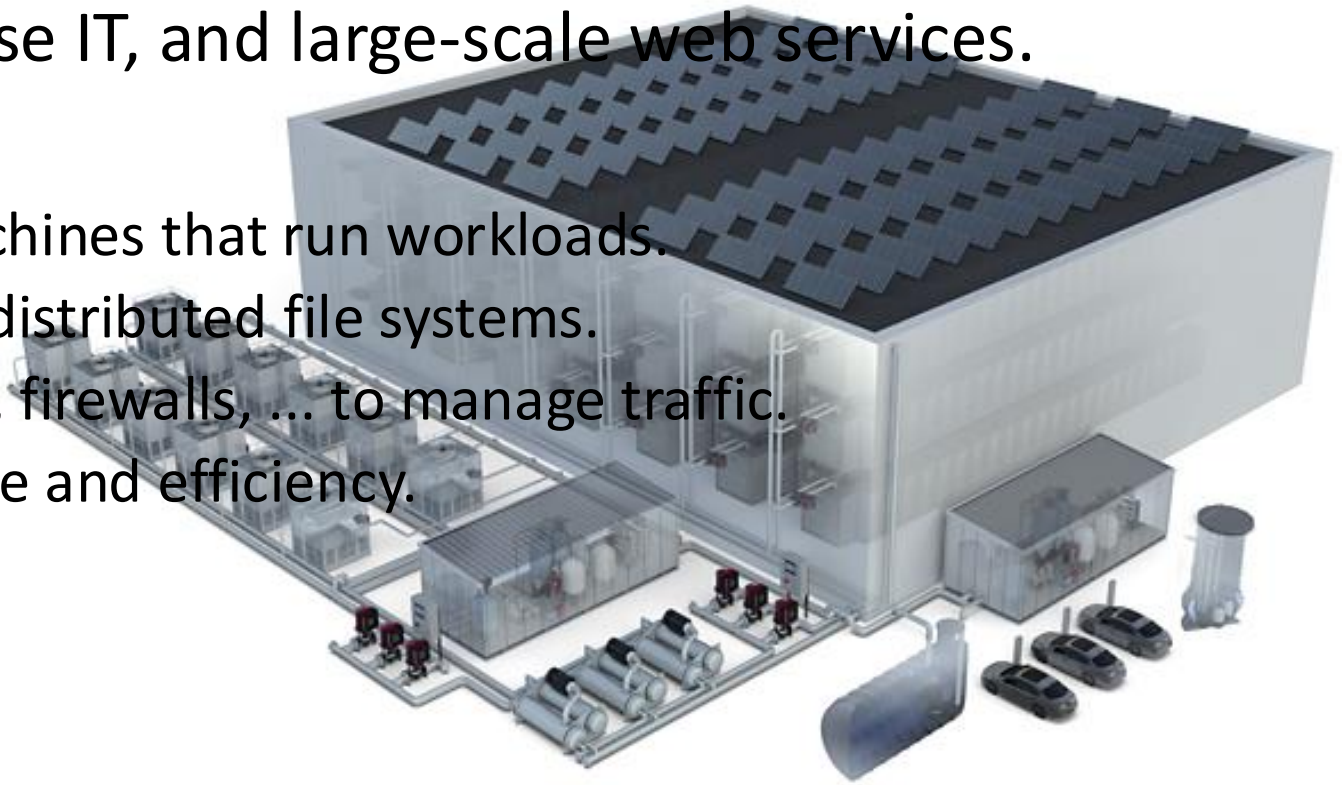
	t1	t2	t3	t4	max
s1	7	8	6	10	
s2	5	4	13	14	
s3	7	4	3	7	
	0	0	0	0	
max					

Task-Server Assignment in Data Centers

Data centers rely on optimal task-server assignments to balance workloads, minimize latency, and reduce energy costs. Algorithms like the Hungarian method match tasks to servers based on resource availability and performance goals. This ensures efficient cloud operations for providers like AWS and Google.

Data centers (DCs)

- Data center ? A facility that houses computing infrastructure (servers, storage, networking equipment) to support applications, services, and data processing. Modern data centers power cloud computing (e.g., AWS, Google Cloud), enterprise IT, and large-scale web services.
- Key Components of DC ?
  - **Servers**: Physical or virtual machines that run workloads.
  - **Storage**: Hard drives, SSDs, or distributed file systems.
  - **Networking**: Switches, routers, firewalls, ... to manage traffic.
  - **Power/Cooling**: Ensures uptime and efficiency.



Task-server assignment Optimization

- **Task-server assignment** is the process of matching workloads or tasks (batch jobs, real time services requests) to servers in a data center to optimize:
  - Performance (minimize latency, maximize throughput).
  - Resource utilization (CPU, RAM, bandwidth).
  - Energy efficiency (reduce power consumption).
- **Why** ?
  - Google saved **40% energy** via task-server optimizations.
  - Netflix avoiding buffering by assigning streams to optimal servers.

Example

- A cloud provider needs to assign 4 tasks (T1–T4) to 4 servers (S1–S4) in a data center. Each task has different CPU/RAM requirements, and each server has limited resources. The goal is to minimize total energy consumption while avoiding overload.
- In this situation The "cost" could represent:
  - **Energy consumption** (watts) if a task runs on a server.
  - **Latency** (ms) between the task's user and the server.
- The implementation of the Hangarian Algorithm will give the result:
  - T1→S1 (5w), T2→S4 (3W), T3→S3 (1W), T4→S2 (6W)
  - **Total Energy = 5 + 3 + 1 + 6 = 15W** (Minimal possible)

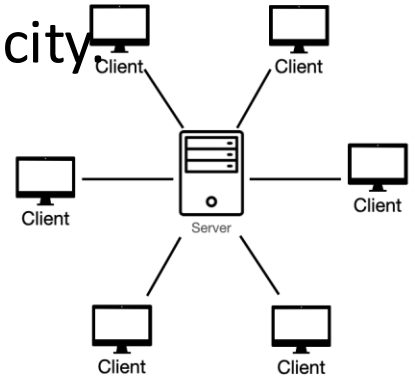
Task \ Server	S1	S2	S3	S4
T1	5	8	6	7
T2	7	4	9	3
T3	3	2	1	5
T4	2	6	4	9

Peer-to-Peer File Download Optimization

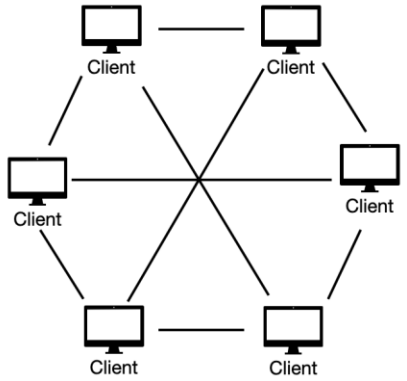
Peer-to-Peer (P2P) networks decentralize file sharing, but optimizing downloads requires strategic chunk-to-peer assignments. While heuristics like "rarest-first" dominate, combinatorial methods (e.g., Hungarian Algorithm) theoretically minimize download times by prioritizing high-bandwidth peers.

Peer-to-Preer (P2P) networks

- **P2P network ?** A decentralized system where devices "peers" directly share resources (e.g., files) without relying on a central server. Each peer acts as both a client (requesting data) and a server (providing data). BitTorrent (file sharing) is a famous example of P2P application.
- **Key Features of P2P:**
  - **Decentralization**: No central authority—peers communicate directly.
  - **Self-Organization**: Peers dynamically join/leave the network.
  - **Scalability**: More peers improve network capacity.
  - **Resilience**: No single point of failure.



Client Server Architecture



P2P Architecture

File Download Optimization

- In P2P file sharing, a file is split into **chunks** (segments). To optimize downloads, peers must **assign chunks to other peers** strategically to:
  - Minimize **download time**.
  - Maximize **throughput** (using high-bandwidth peers).
  - Avoid redundant downloads (e.g., fetching the same chunk from multiple peers).
- **In real systems:** The Hungarian Algorithm (or other combinatorial methods) is used but P2P networks like BitTorrent often use heuristics.

Example

- In a P2P network a peer (P0) would to download file split into 4 chunks (C1, C2, C3, C4) from 4 peers (P1, P2, P3, P4) with varying bandwidth and chunk availability. The goal is to assign chunks to peers to minimize total download time for a new peer (P0).
- In this situation The "cost" represents the time to download a chunk from a peer (based on bandwidth and latency).
- The implementation of the Hangarian Algorithm will give the result:
  - C1→P1 (2 sec), C2→P3 (2 sec), C3→P4 (5 sec), C4→P2.
  - **Total Time**: Dominated by the slowest chunk (5 sec).

Chunk \ Peer	P1	P2	P3	P4
C1	2	5	∞	3
C2	4	3	2	∞
C3	∞	6	4	5
C4	3	∞	5	2