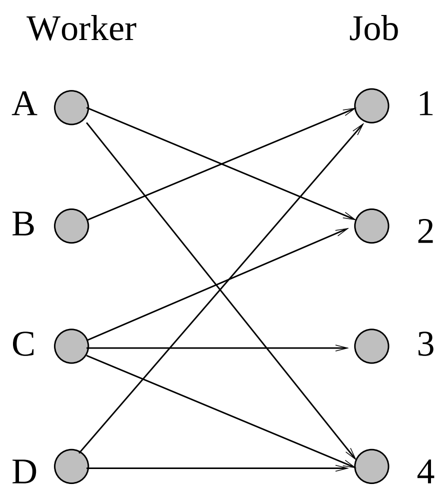
Assignment and Matching

1. **A matching is a pairing of nodes—*collection of disjoint edges*.**



2. **Bipartite graph. Two node classes, workers and jobs.**

3. **An edge (i, j) means worker i can do job j.**

4. **If weighted, then $c(i, j)$ is the *proficiency* of i at job j. (In unweighted case, $c(i, j) = 1$.)**

5. **Workers to jobs assignment for *maximizing total proficiency*.**

6. **Each worker assigned to at most one job and vice versa, so this is a matching.**

Applications

1. **[Rooming Problem.] Dorm room assignment. Graph G with students as nodes. Weight c_{ij} is *compatibility* of pair (i, j).**

2. **[Airline Pilot Assignment.]**

   - **Airlines need to form teams of captain and first officer.**
   - α_i **is effectiveness of i as captain.**
   - β_i **is effectiveness of i as 1st officer.**
   - **Seniority Rule: captain more senior.**
   - **Make edge weight**

$$c_{ij} = \begin{cases} \alpha_i + \beta_j & \text{if } i \text{ more senior} \\ \alpha_j + \beta_j & \text{otherwise} \end{cases}$$

3. **In these applications, the graph is *not* bipartite. We will only study the bipartite case.**

More Applications

4. [Stable Marriage.]

- Men $\{A, B, \ldots, Z\}$, women $\{a, b, \ldots, z\}$.
- Their preference tables.

Men's Preferences

A	b	c	a
B	b	a	c
C	c	a	b

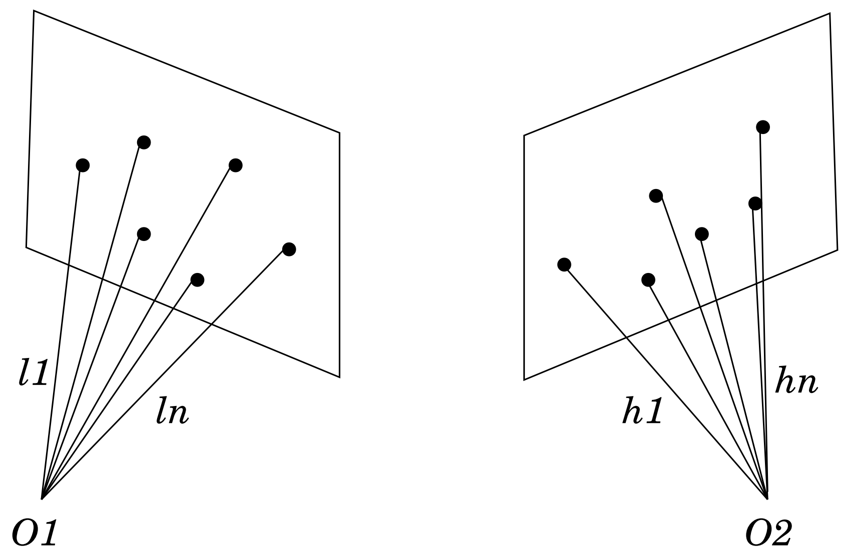
Women's Prefereces

a	C	B	A
b	A	C	B
c	A	C	B

- A matching M.
- M is unstable if $\exists$ pair $(Bob, Sally)$ who like each other more than their spouses.
- Is stable marriage always possible?
- Medical schools use this protocol.
- Gale-Shapely Theorem: A stable marriage always possible, and found in $O(n^2)$ time.

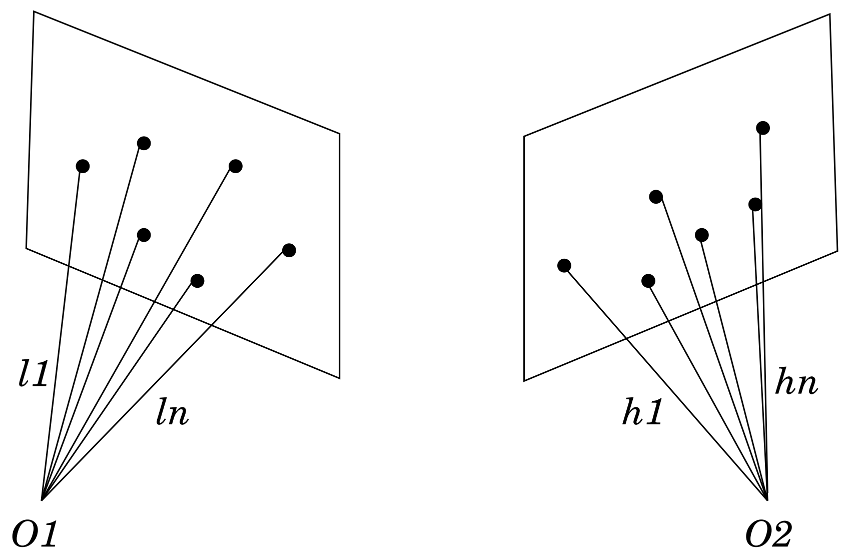
Stereo Vision

1. **Stereo matching to locate objects in space.**
2. **Infrared sensors at two different locations.**
3. **Each sensor gives the angle of sight (line) on which the object lies.**



4. **If p objects, we get two sets of lines: $\{L_1, L_2, \ldots, L_p\}$ and $\{L'_1, L'_2, \ldots, L'_p\}$.**

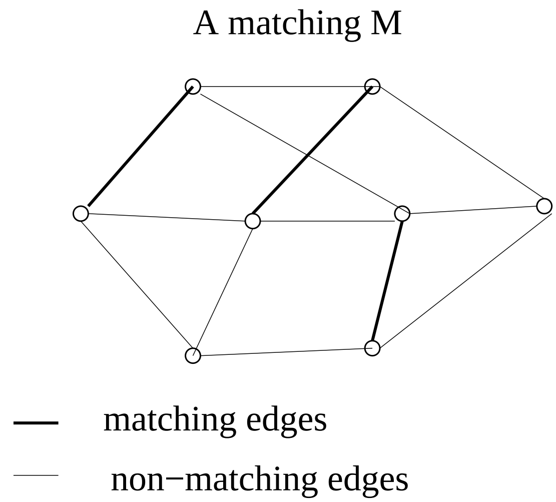
Stereo Vision



1. **Two problems: (1) a line from one sensor might intersect multiple lines from the other; (2) due to noise, the lines for the same object may not intersect.**
2. **Solve the problem using assingment. Nodes are lines. Cost c_{ij} is the distance between L_i and L'_j.**
3. **Distance between lines of the same object should be close to zero.**
4. **Optimal assingment should give excellent matching of line.**

Definitions

1. **A** *matching* $M \subseteq E$**, in graph** $G = (V, E)$**, is a set of edges no two sharing a vertex.**



2. $|M|$ **is the** *cardinality* **of** M**.**

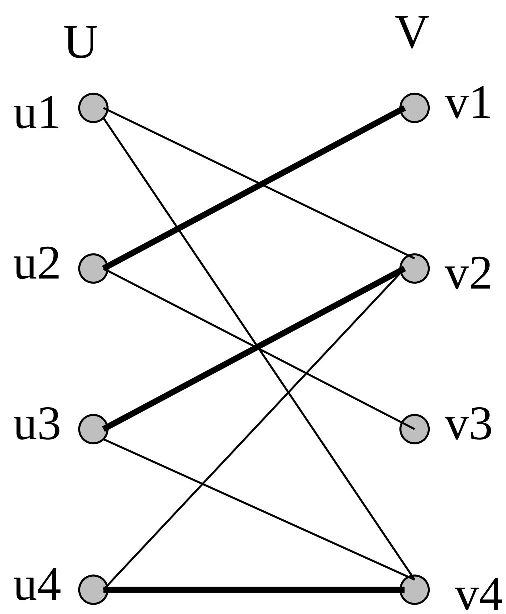
3. **In unweighted graphs, find max cardinality matching.**

4. **In weighted graphs, find max weight matching.**

5. **A matching is perfect if all vertices are matched.**

Perfect Matching

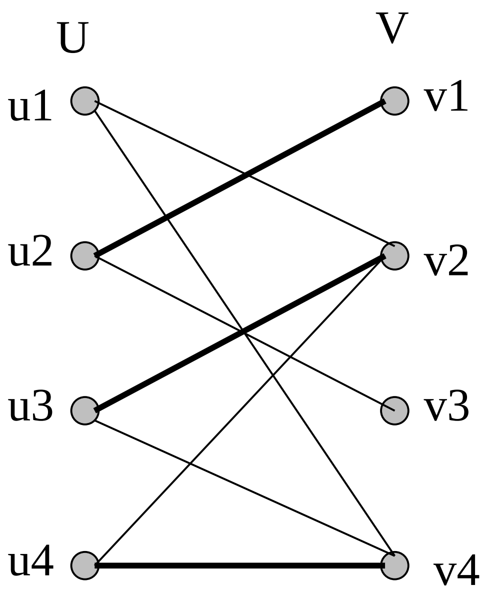
1. **Consider a bipartite graph $G = (U, V, E)$.**
2. **When does G have a perfect matching?**



Neighborhoods

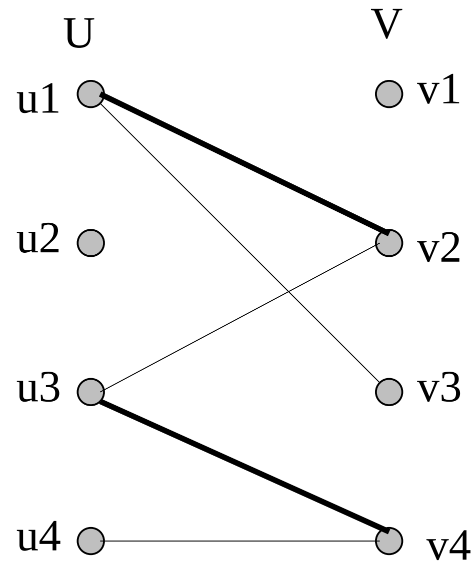
1. **A subset $S \in U$.**
2. **Neighborhood $N(S)$ is vertices of V adjacent to any vertex in S.**
3. **For example, $N(u_1) = \{v_2, v_4\}$.**

Hall's Theorem: G has a perfect matching iff $|N(S)| \geq |S|$, for all $S \subseteq U$.



Alternating Paths

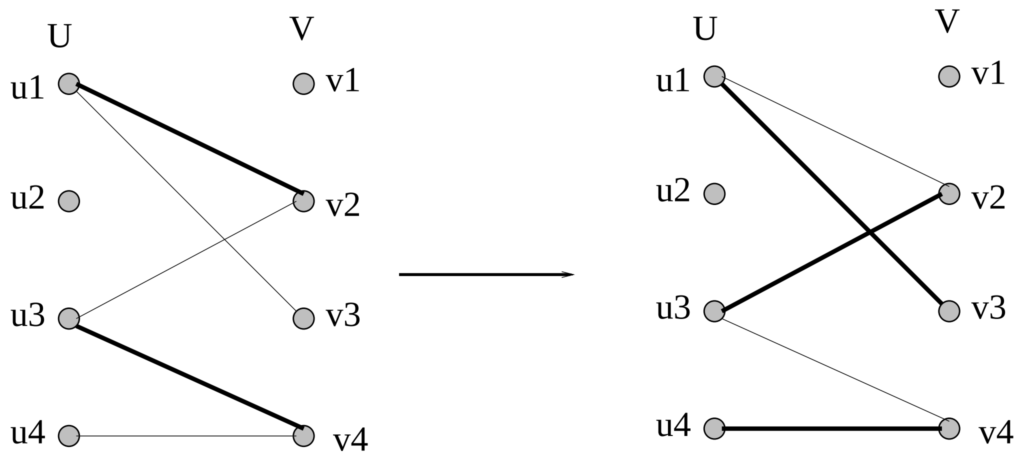
1. **A matching M has some *matched* and some *unmatched* (free) vertices.**
2. ***Alternating path* has edges alternating between M and $E - M$.**



3. **u_2, u_4 are free, while u_1, u_3 are matched.**
4. **Path $u_4, v_4, u_3, v_2, u_1, v_3$ is alternating.**
5. **An alternating path is *augmenting* if both of its endpoints are *free* vertices.**
6. **Path $u_4, v_4, u_3, v_2, u_1, v_3$ is also augmenting.**

Augmenting Matching

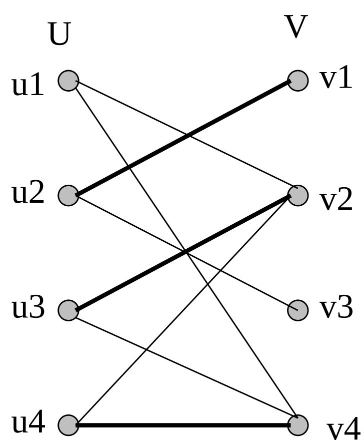
1. **If a matching M has an augmenting path, then we get a larger matching M' by swapping the edges on the augmenting path.**



Hall's Theorem

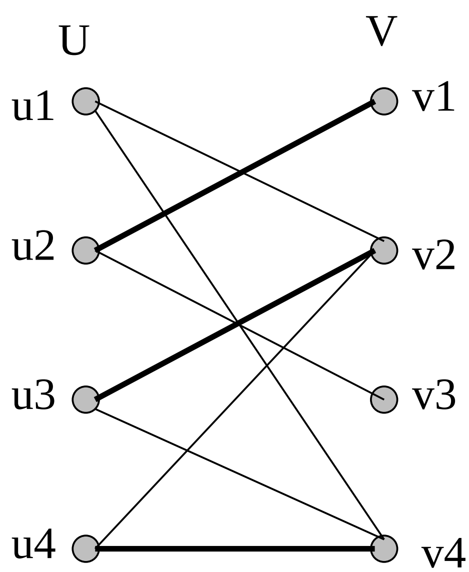
Hall's Theorem: G has a perfect matching iff $|N(S)| \geq |S|$, for all $S \subseteq U$.

1. **The direction PM $\Rightarrow$ $|N(S)| \geq |S|$ is easy.**
2. **Consider any set $S \subseteq U$.**
3. **Let $mate(u)$ be the vertex matched with u.**
4. **Since $mate(u_i) \neq mate(u_j)$, it follows that $| \cup_{u \in S} \; mate(u)| \geq |S|$.**
5. **Since $mate(u) \in N(u) \subseteq V$, we must have $|N(S)| \geq |S|$.**

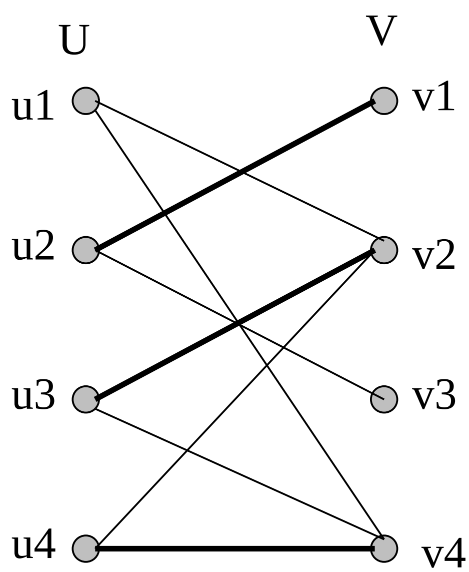


Hall's Proof

1. **Now, suppose $|N(S)| \geq |S|$ holds, for all $S \subseteq U$.**
2. **Consider a max matching M, and let u be a free vertex in it.**
3. **Let Z be the set of all vertices *reachable* from u with an *alternating* path.**
4. **There is no free vertex in Z (except u)–otherwise, an augmenting path, which contradicts M's max size.**
5. **Let $L = Z \cap U$, and $R = Z \cap V$.**



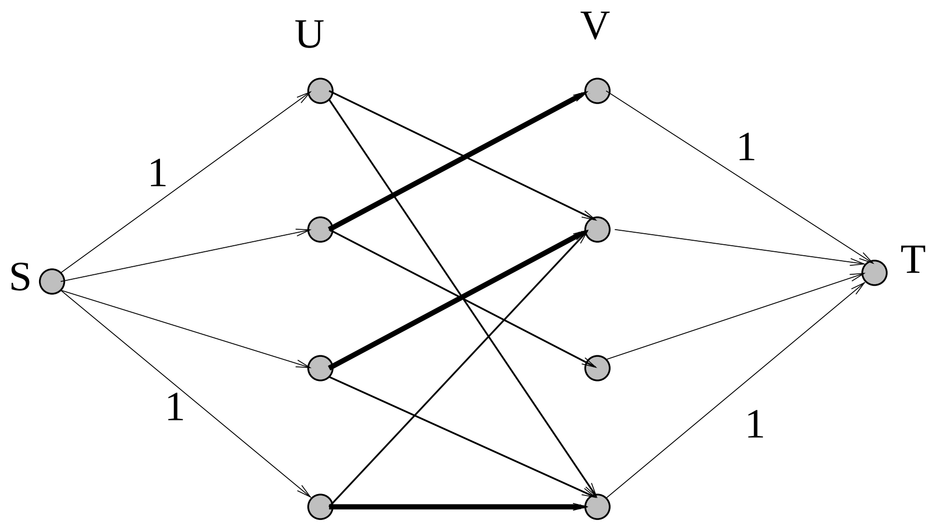
Hall's Proof



1. **Observe that** $N(L) = R$**.**
2. **Each vertex in** L **(except** u**) is matched with someone in** R**, and the mates are distinct.**
3. **So,** $|R| = |L| - 1$**.**
4. **But then** $|N(L)| < |L|$**, a contradiction!**

Unweighted Bipartite Matching

1. **Unweighted matching via maxflow.**



All capacities 1

2. **Maximum flow value equals $|M|$.**

3. **With integral flow,** *matching* $\Leftrightarrow$ *flows*.

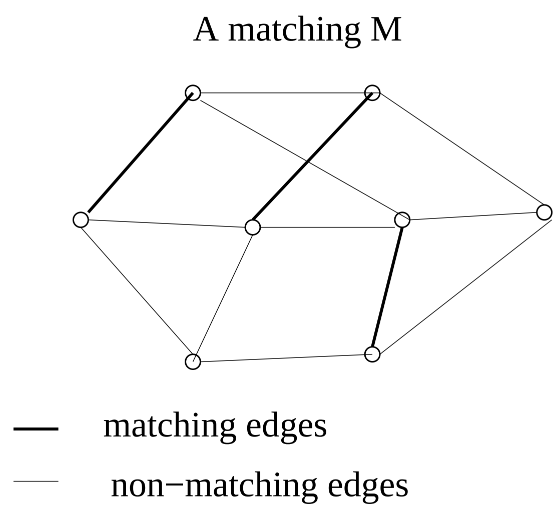
4. **With integer capacities, there always exists an integral flow. (Why?)**

5. **Think Ford-Fulkerson algorithm.**

6. **Max cardinality matching solved in $O(n^3)$ time.**

Remarks

1. **No such transformation for non-bipartite matching.**



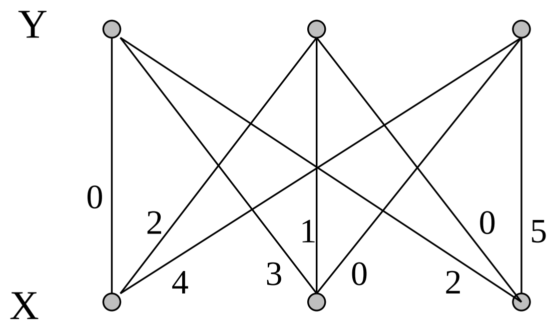
2. **Actually, Hall's Theorem also invalid for general graphs. (Example: a triangle.)**

   Tutte's Theorem: *G has a perfect matching iff for all $S \subseteq V$, $oc(G - s) \leq |S|$, where oc is number of odd-cardinality components.*

3. **For bipartite matching, specialized maxflow algorithm for unit networks runs in $O(\sqrt{n}m)$. (Read Section 8.2.)**

Hungarian Method for Assignment

- **Maxflow method does not work for weighted matching.**
- $G = (X, Y, E)$**, with edge weights** $w(e)$**, which is weight or benefit of** e**.**
- **Find optimal (max weight) assignment.**



- **Assume complete graph—missing edges given** $w(e) = 0$**. So, want a max weight** *perfect* **matching in** G**.**

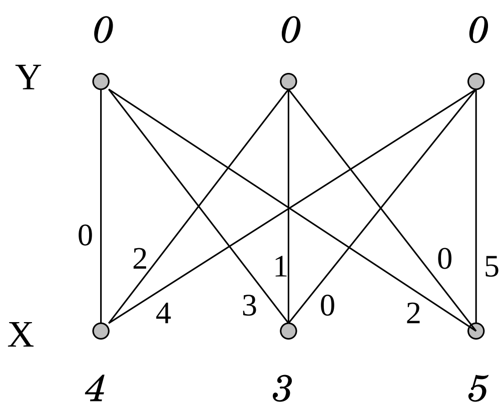
Feasible Vertex Labeling

- **Real-valued labels $\ell()$ such that**

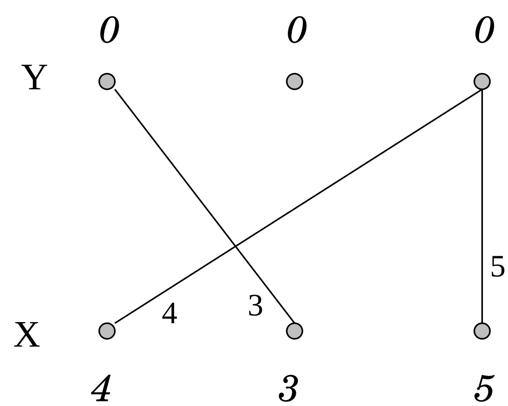
$$\ell(x) \; + \; \ell(y) \; \geq \; w(x,y), \quad \forall x \in X, y \in Y.$$

- **Initial feasible labeling:**

$$\begin{aligned} \ell(x) &= \max_{y \in Y}\{w(x,y)\} \quad \textbf{for } x \in X \\ \ell(y) &= 0 \quad \textbf{for } y \in Y \end{aligned}$$



Vertex Labeling



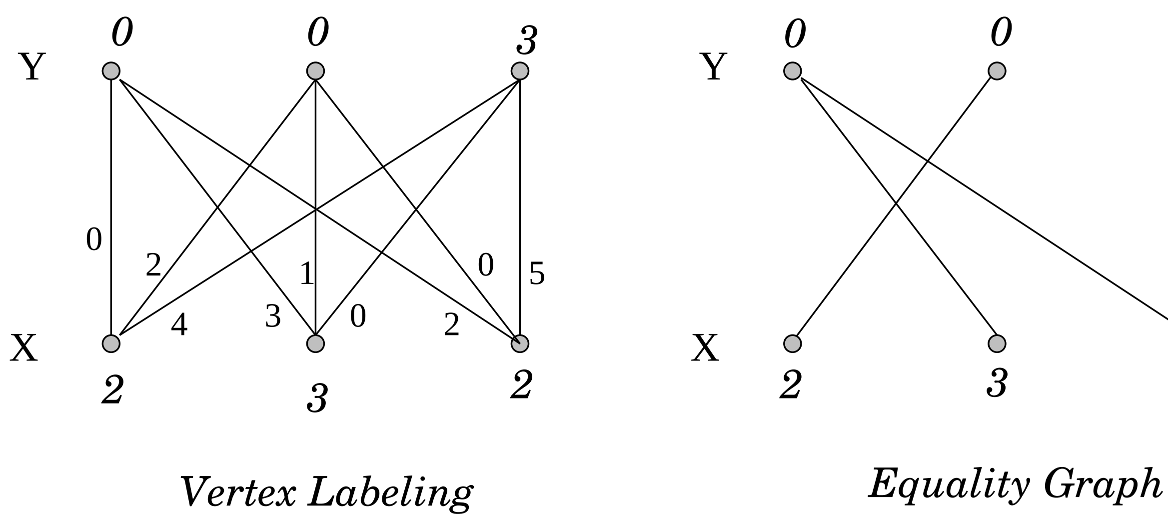
Equality Graph

- **[Equality Graph]** $G_\ell = (X, Y, E_\ell)$**, where**

$$E_\ell \; = \; \{(x,y) \mid \ell(x) + \ell(y) = w(x,y)\}.$$

Main Theorem

[Kuhn-Munkres.] *For any feasible labeling ℓ, if G_ℓ contains a perfect matching M, then M is an optimal assignment.*



1. **In a PM, each vertex is covered exactly once, so** $w(M) = \sum_{e \in M} w(e) = \sum_{v \in V} \ell(v)$
2. **Any other assingment** M' **in** G **satisfies** $w(M') = \sum_{e \in M'} w(e) \leq \sum_{v \in V} \ell(v)$
3. **Thus,** $w(M') \leq w(M)$**, and** M **must be optimal.**

Hungarian Method

1. **Initialize vertex labeling ℓ. Determine G_ℓ.**

2. **Pick any matching M in G_ℓ.**

3. **If M perfect, stop. Otherwise, pick a free vertex $u \in X$. Set $S = \{u\}$, and $T = \emptyset$.**

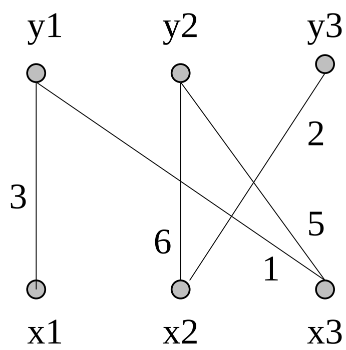
4. **If $N(S) = T$, update labels:**
   $\alpha_\ell = \min_{x \in S,\ y \notin T} \{\ell(x) + \ell(y) - w(x, y)\}$

$$\ell'(v) = \left\{ \begin{array}{ll} \ell(v) - \alpha_\ell & \textbf{if } v \in S \\ \ell(v) + \alpha_\ell & \textbf{if } v \in T \\ \ell(v) & \textbf{otherwise} \end{array} \right\}$$

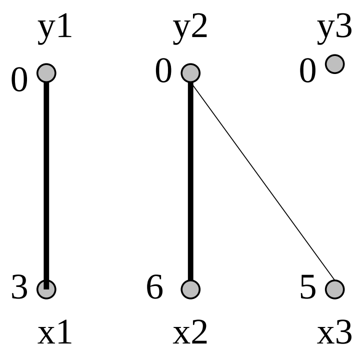
   (Now $N(S) \neq T$.)

5. **If $N(S) \neq T$, pick $y \in N(S) - T$.**
   - **If y free, u–y is augmenting path; augment M and go to 3.**
   - **If y matched, say, to z, then extend alternating tree: $S = S \cup \{z\}$, $T = T \cup \{y\}$. Go to 4.**

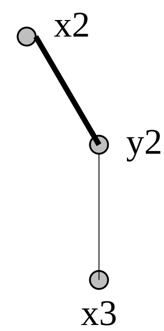
Example of Hungarian



Initial Graph



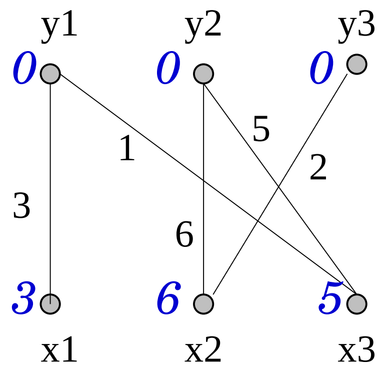
Equality Graph
Matching M

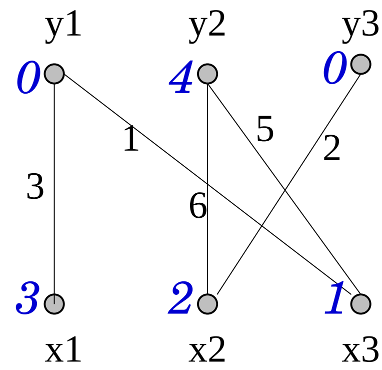


Alternating Tree
wrt M

- **A 3×3 assignment problem.**
- **Initial labels and equality graph.**
- **Initial matching $(x_1, y_1), (x_2, y_2)$.**
- $S = \{x_3\}$**,** $T = \emptyset$**.**
- **Since $N(S) \neq T$, we do step 5. Choose $y_2 \in N(S) - T$.**
- y_2 **matched, add** $y_2 x_2$ **(grow tree).**
- **Since $N(S) = T$, do step 4.**

Example contd.





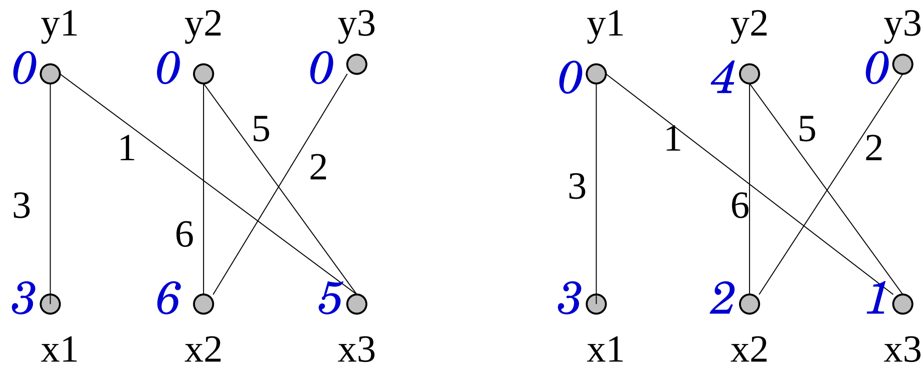
New Eq. Graph

- $S = \{x_2, x_3\}$, **and** $T = \{y_2\}$.
- **Calculate** α**:**

$$\alpha_\ell = \min_{x \in S, y \notin T} \begin{cases} 6+0-0 & x_2 y_1 \\ 6+0-2 & x_2 y_3 \\ 5+0-1 & x_3 y_1 \\ 5+0-0 & x_3 y_3 \end{cases}$$
$$= 4$$

- **Reduce labels for** S**, increase for** T**, by 4.**
- **New equality graph has a perfect matching.**

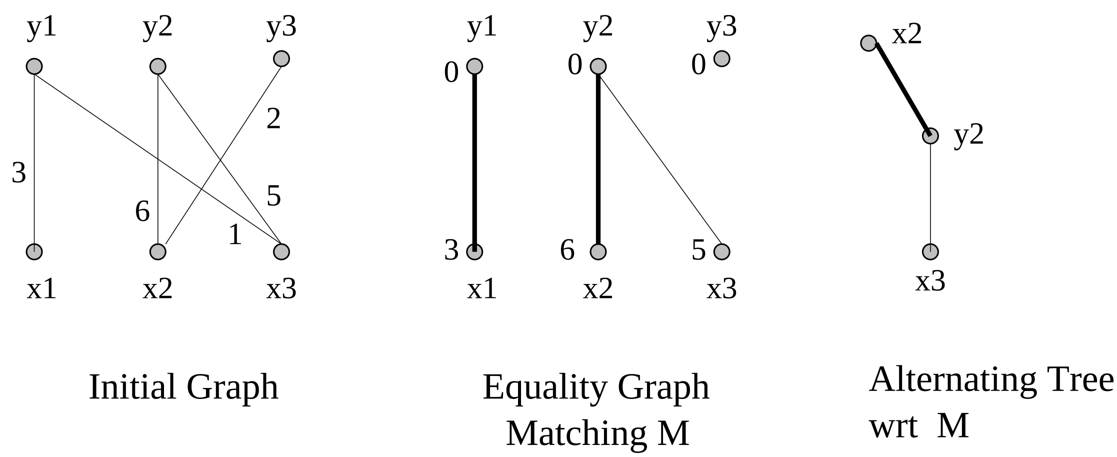
Analysis



New Eq. Graph

- **Relabeling ensures that at least one new edge added to G_ℓ.**
- **Relabeling ensures no edge of G_ℓ removed.**
- **In a worst-case, all edges of G would eventually appear in G_ℓ.**
- **Thus, a perfect matching guaranteed to be found.**

Complexity



- **Algorithm has n phases, in each phase matching size grows by 1.**
- **Keep track of edge with smallest slack:** $\ell(x) + \ell(y) - w(x, y)$**, where** $x \in S, y \notin T$**.**
- **Initial slack calculation takes** $O(n^2)$ **time.**
- **When a vertex moves from** $\bar{S}$ **to** S**, we compute slacks for all** $y \notin T$**.**
- **In each phase, at most** n **vertices go from** $\bar{S}$ **to** S**, so** n **slack re-calculations, each at** $O(n)$ **time, for a total of** $O(n^2)$**.**
- **Total algorithm takes** $O(n^3)$**.**

Other Results on Matching

- **[Bipartite Cardinality Matching:]** $O(\sqrt{n}m)$ **time. Maxflow on unit capacity networks. [Even-Tarjan]**
- **[Non-Bipartite Cardinality Matching:]** **First polynomial time,** $O(n^4)$**, algorithm in 1957 by Edmonds.**
  Current best $O(\sqrt{n}m)$ **[Micali-Vazirani].**
- **[Bipartite Weighted Matching:]** $O(nm + n^2 \log n)$ **strongly poly.**
  $O(\sqrt{n}m \log(nC))$ **scaling algorithm.**
- **[Non-Bipartite Weighted Matching:]** $O(n^3)$ **by Edmonds+Gabow '75.**
  Current best $O(nm + n^2 \log n)$**.**

Stable Matching Problem

- **Society of n men $(A, B, \ldots, Z)$ and n women $(a, b, \ldots, z)$.**
- **Each man (woman) ranks all women (man), in descending order of preference.**

Men's Preferences

A	c	a	b
B	c	b	a
C	a	b	c

Women's Preferences

a	C	B	A
b	B	A	C
c	B	C	A

- **A matching is a 1-to-1 correspondence (monogamous, heterosexual marriage).**
- **A pair (M, w) is *unstable* if M and w like each other more their assigned partners.**
- **A matching is called unstable if it has a unstable pair (risks elopement).**
- **Determine a stable matching.**

Stable Matching

- **The matching $\{(A,c),(B,b),(C,a)\}$ leads to an unstable pair (B,c).**

Men's Preferences

A	c	a	b
B	c	b	a
C	a	b	c

Women's Preferences

a	C	B	A
b	B	A	C
c	B	C	A

- **The matching $\{(A,b),(B,c),(C,a)\}$ is stable.**

Applications:

- **The method used by Medical Schools for selecting residents.**
- **Hong Kong state universities use stable matching for admissions.**
- **Several books just on stable marriage.**

Stable Matching Theorem

Theorem: Stable matching is always possible. [Gale Shapely 1955]

- **We will prove this theorem by presenting an algorithm that always returns a stable matching.**
- **Basic principle:** *Man proposes, woman disposes.*
- **Each unattached man proposes to the highest-ranked woman in his list,** *who has not already rejected him.*
- **If the man proposing to her is better than her current mate, the woman dumps her current partner, and becomes engaged to the new proposer.**
- **Since no man proposes to the same woman twice, the algorithm terminates, and we prove the result is a stable matching.**

Stable Matching Algorithm

- $LIST$: **list of unattached men.**
- $cur(m)$: **highest ranked woman in m's list, who has not rejected him.**
- **Initialize** $LIST = \{1, 2, \ldots, n\}$ **and** $cur(i) = M(i, 1)$.
- **Choose a man, say, Bob, from** $LIST$. **Bob proposes to Alice, where Alice** $= cur$**(Bob).**
- **If Alice unattached, Bob and Alice are engaged.**
- **If Alice is engaged to, say, John, but prefers Bob, she dumps John, and Bob and Alice are engaged. Otherwise, she rejects Bob.**
- **The rejected man rejoins** $LIST$, **and updates his** cur.
- **Output the engaged pairs when** $LIST = \emptyset$.

Illustration of the Algorithm

Men's Preferences

A	c	a	b
B	c	b	a
C	a	b	c

Women's Preferences

a	C	B	A
b	B	A	C
c	B	C	A

- **Final matching** $\{(A, b), (B, c), (C, a)\}$**.**
- **The algorithm terminates in** $O(n^2)$ **steps, since each step moves one** cur **pointer, and there are at most** n^2 **preferences.**
- **Remains to prove the matching is always stable.**

Correctness

- **Suppose the resulting matching has a unstable pair (Dick, Laura).**
- **Dick must have proposed to Laura at some point.**
- **During the algorithm, Laura also rejected Dick in favor of some she prefers more.**
- **Since no woman ever switches to a man less desirable than her current partner, Laura's current partner must be more desirable than Dick.**
- **Thus, the pair (Dick, Laura) is not unstable.**