Independent Sets, Covers and Maximum Bipartite Matching

Dr. Rajiv Misra

Associate Professor

Dept. of Computer Science & Engg.

Indian Institute of Technology Patna

rajivm@iitp.ac.in

Preface

Recap of Previous Lecture:

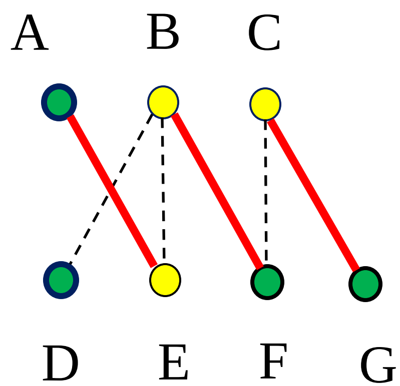
- In the previous lecture, we have discussed the Concept of Matching, Perfect matchings, Maximal matchings, Maximum Matchings, *M*-alternating path, *M*-augmenting path, Symmetric difference, Hall's Matching condition and Vertex covers.

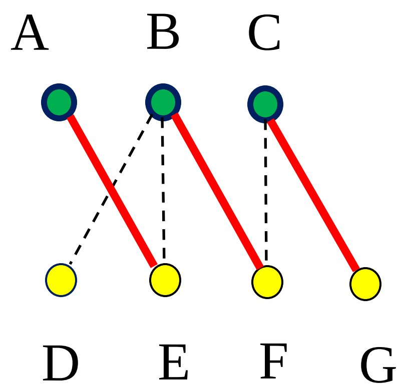
Content of this Lecture:

- In this lecture, we will discuss König-Egerváry theorem, Independent sets, Covers i.e. edge cover, vertex cover, Maximum bipartite matching and Augmenting Path Algorithm.

Theorem: If G is a bipartite graph, then the maximum size of a matching in G equals the minimum size of a vertex cover of G.

Green: Vertex cover
Red: Matching
$|Q| \geq |M|$

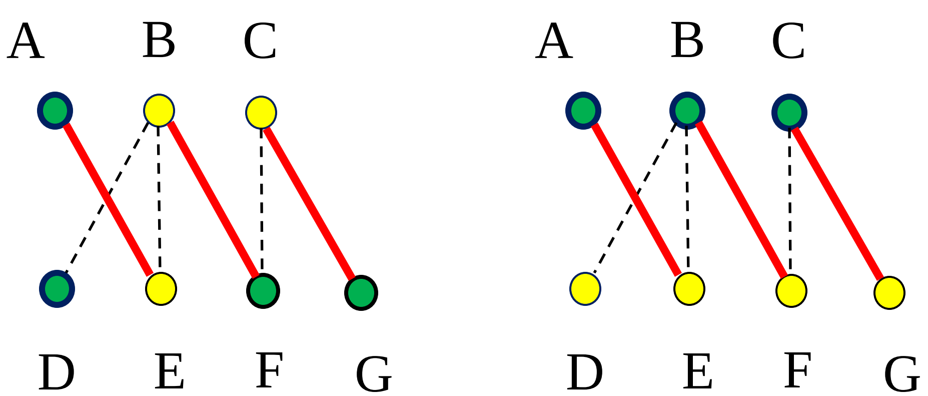




Proof : Let G be an X, Y-bigraph.

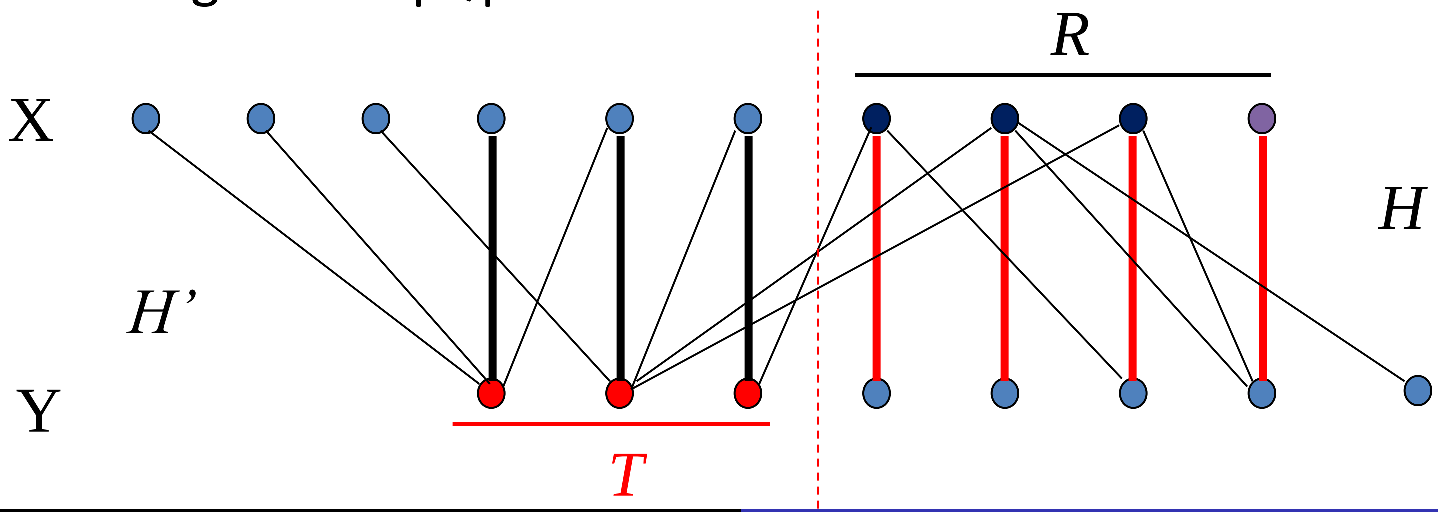
- Since distinct vertices must be used to cover the edges of a matching, $|Q| \geq |M|$ whenever Q is a vertex cover and M is a matching in G.
- Given a smallest vertex cover Q of G, we **construct a matching of size $|Q|$ to prove that equality can always be achieved.**

Green: Vertex cover
Red: Matching
$|Q| \geq |M|$



Theorem 3.1.16 Continue

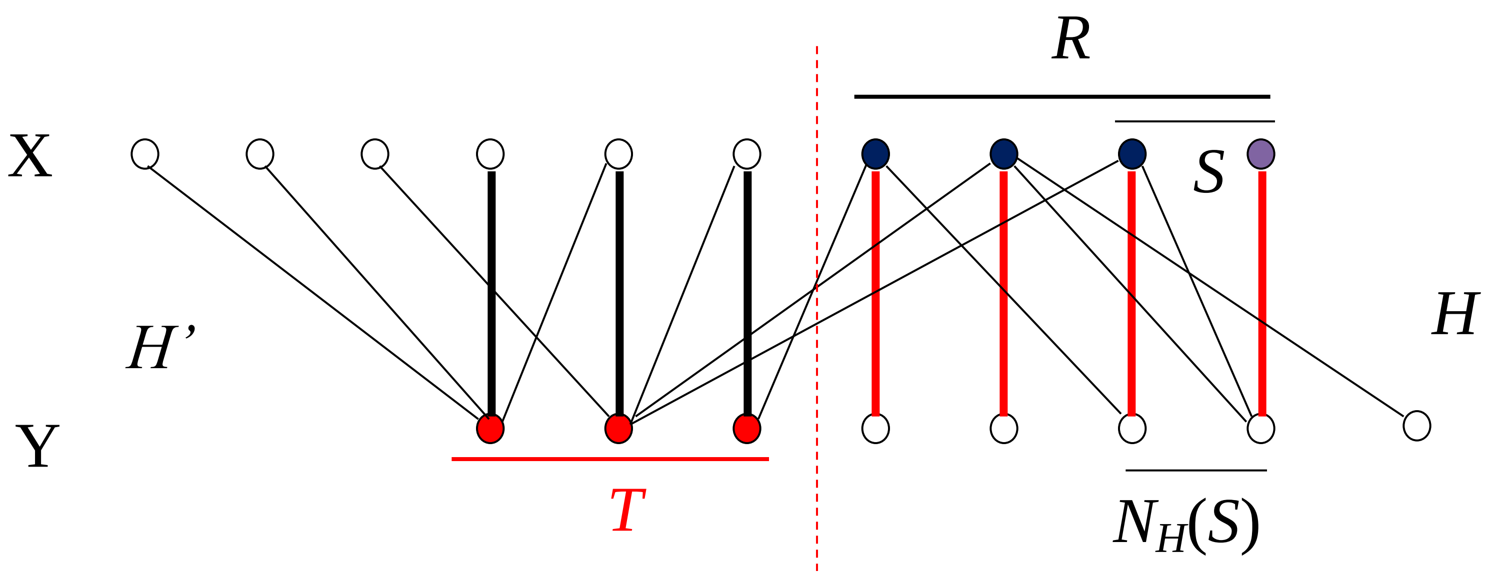
- Partition min vertex cover Q into $R = Q \cap X$ and $T = Q \cap Y$.
  - Let H be the subgraphs of G induced by $R \cup (Y\text{-}T)$ and H' be the subgraphs of G induced by $T \cup (X\text{-}R)$.
  - We use Hall's Theorem to show that H has a matching that saturates R into $Y\text{-}T$ and H' has a matching that saturates *T into* $X\text{-}R$.
  - Since H and H' are disjoint, the two matchings together form a matching of size $|Q|$ in G.



Theorem 3.1.16 Continue

- Since $R \cup T$ is a vertex cover, G has no edge from Y-T to X-R.
- For each $S \subseteq R$, we consider $N_H(S)$, which is contained in Y-T. If $|N_H(S)|<|S|$, then we can substitute $N_H(S)$ for S in Q to obtain a smaller vertex cover, since $N_H(S)$ cover all edges incident to S that are not covered by T.

The minimality of Q thus yields Hall's Condition in H, and hence H has a matching that saturates R. Applying the same argument to H' yields the matching that saturates T.



Remark 3.1.17

- A **min-max relation** is a theorem stating equality between the answers to a minimization problem and a maximization problem over a class of instances.
  - The **Konig-Egervary Theorem** is such a relation for vertex covering and matching in bipartite graphs.
- Consider a pair of **dual optimization problems** as a maximization problem **M** and a minimization problem **N**, defined on the same instances (such as graphs), such that for every candidate solution M to **M** and every candidate N to **N** (on the same instance), the value of M is less than or equal to the value of N. Often the "value" is cardinality, as when **M** is maximum matching and **N** is minimum vertex cover.

Remark continue

- When ***M*** and ***N*** are dual problems, obtaining candidate solutions ***M*** and ***N*** that have the same value PROVES that ***M*** and ***N*** are optimal solutions for that instance.

- A **min-max relation** states that, on same class of instances, these short proofs of optimality exist.

- These theorems are desirable because they save work. Our next objective is another such theorem for independent sets in bipartite graphs.

Duality of Max Matching

- Let X_{ij} indicate whether (i, j) is included in the matching.

$$\max \sum_{i\in L, j\in R} X_{ij} \qquad\qquad \min \sum_{i\in L} Y_i + \sum_{j\in R} Y_j$$

$$s.t. \sum_{i\in L} X_{ij} \le 1 \ \forall j \in R \qquad Y_j \qquad\qquad s.t. Y_i + Y_j \ge 1 \quad \forall i \in L, j \in R$$

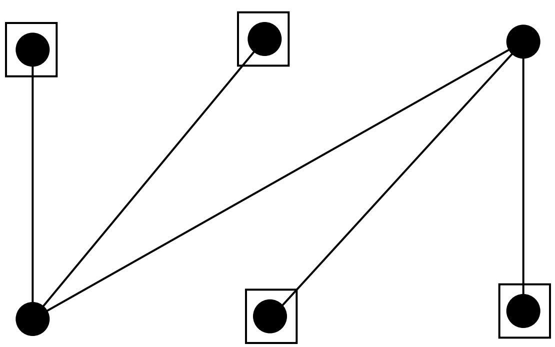
$$\sum_{j\in R} X_{ij} \le 1 \ \ \forall i \in L \qquad Y_i \qquad\qquad Y_i, Y_j \ge 0 \ \ \forall i \in L, j \in R$$

$$X_{ij} \ge 0$$

Independent sets and Covers

- The **independence number** of a graph is the maximum size of an independent set of vertices.
- The **independence number** of a bipartite graph does ***not*** always equal the size of a partite set.

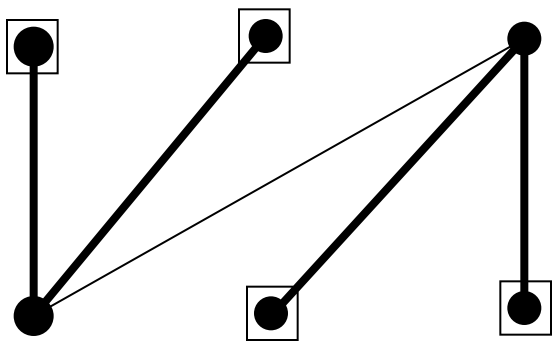
  - **Example:** In the graph below, both partite sets have size 3, but we have marked an independent set of size 4.



No vertex covers two edges of a matching. Similarly, no edge contains two vertices of an independent set. This yields another dual covering problem.

Edge cover 3.1.19

- An ***edge cover*** of G is a set L of edges such that every vertex of G is incident to some edge of L.
- **Example:** We say that the vertices of G are ***covered*** by the edges of L. In the given example, the four edges incident to the marked vertices form an edge cover, the remaining two vertices are covered "for free."



- Only graphs without isolated vertices have edge covers. A perfect matching forms an edge cover with ***n(G)*/2 edges**. In general, we can obtain an edge cover by adding edges to a maximum matching.

Definitions

- For the optimal sizes of the sets in the independence and covering problems, following notations are defined:
    - Maximum size of independent set $\alpha(G)$
    - Maximum size of matching $\alpha'(G)$
    - Minimum size of vertex cover $\beta(G)$
    - Minimum size of edge cover $\beta'(G)$

Notations

- A graph may have many independent sets of maximum size (C_5 has five of them), but the independence number $\alpha(G)$ is a single integer ($\alpha(C_5)=2$).

- The notation treats the numbers that answer these optimization problems as graph parameter, like the order, size, maximum degree, diameter, etc. The use of $\alpha'(G)$ to count the edges in a maximum matching suggests a relationship with the parameter $\alpha(G)$ that counts the vertices in a maximum independent set.

Contd...

- In this notation, the König-Egerváry theorem states that $\alpha'(G) = \beta(G)$ for every bipartite graph G.

- We will prove that also $\alpha(G) = \beta'(G)$ for bipartite graphs without isolated vertices. Since no edge can cover two vertices of an independent set, the inequality $\beta'(G) \geq \alpha(G)$ is immediate.

- (When $S \subseteq V(S)$, we often use $\overline{S}$ to denote $V(G)$-S, the remaining vertices)

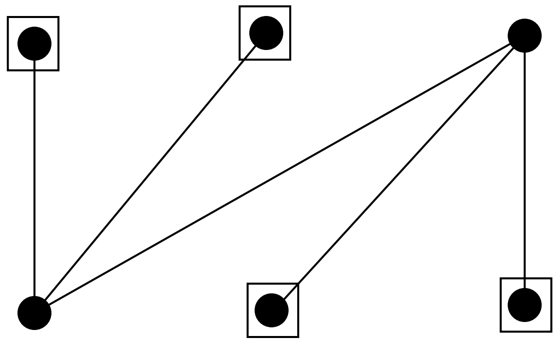
Lemma: In a graph G, $S \subseteq V(G)$ is an independent set if and only if $\overline{S}$ is a vertex cover, and hence $\alpha(G) + \beta(G) = n(G)$.

Proof :

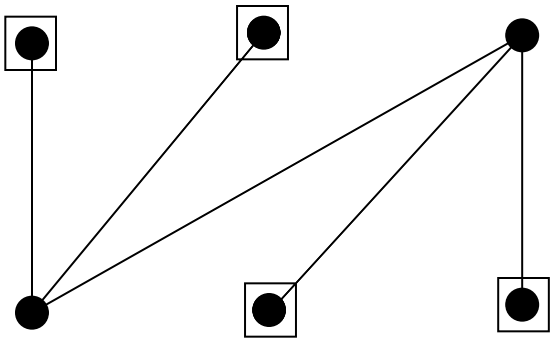
- If S is an independent set, then every edge is incident to at least one vertex of $\overline{S}$
- Conversely, if $\overline{S}$ covers all the edges, then there are no edges joining vertices of S.

Hence every maximum independent set is the complement of a minimum vertex cover, and $\alpha(G) + \beta(G) = n(G)$

Independent set



Lemma: In a graph G, $S \subseteq V(G)$ is an independent set if and only if $\bar{S}$ is a vertex cover, and hence $\alpha(G) + \beta(G) = n(G)$.



$$\alpha(G) = 4$$
$$\beta(G) = 2$$
$$N(G) = 6$$

- The relationship between matchings and edge coverings is more subtle. Nevertheless, a similar formula holds.

Proof:

- From a maximum matching M, we will construct an edge cover of size $n(G)-|M|$.
  - Since a smallest edge cover is no bigger than this cover, this will imply that $\beta'(G) \leq n(G) - \alpha'(G)$.
- Also, from a minimum edge cover L, we will construct a matching of size $n(G) - |L|$.
  - Since a largest matching is no smaller than this matching, this will imply that $\alpha'(G) \geq n(G) - \beta'(G)$.
- These two inequalities complete the proof.

Corollary 3.1.24

- **(König [1916])** If G is a bipartite graph with no isolated vertices then $\alpha(G) = \beta'(G)$.

- **Proof:** By Lemma 3.1.21 and Theorem 3.1.22, $\alpha(G) + \beta(G) = \alpha'(G) + \beta'(G)$. Subtracting the König-Egerváry relation $\alpha'(G) = \beta(G)$ completes the proof.

Maximum Bipartite Matching 3.2

- To find a maximum matching, we iteratively seek augmenting paths to enlarge the current matching.

- In a bipartite graph, if we don’t find an augmenting path, we will find a vertex cover with the same size as the current matching, thereby proving that the current matching has maximum size.

- This yields both an algorithm to solve the maximum matching problem and an algorithmic proof of the König-Egerváry theorem.

Maximum Bipartite Matching Continue

- Given a matching M in an X, Y –bigraph G, we search for M-augmenting paths from each M-unsaturated vertex in X. We need only search from vertices in X, because every augmenting path has odd length and thus has ends in both X and Y.

- We will search from the unsaturated vertices in X simultaneously. Starting with a matching of size 0, $\alpha'(G)$ applications of the Augmenting Path Algorithm produce a maximum matching.

Algorithm: Augmenting Path Algorithm 3.2.1

Input: An X, Y-bigraph G, a matching M in G, and the set U of M-unsaturated vertices in X.

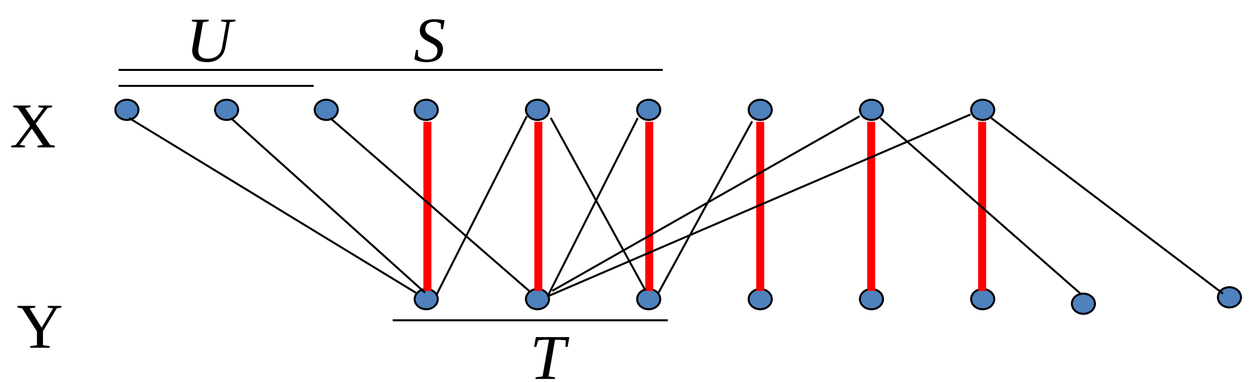
Idea: Explore M-alternating paths from U, letting $S \subseteq X$ and $T \subseteq Y$ be the sets of vertices reached. *Mark* vertices of S that have been explored for path extensions. As a vertex is reached, record the vertex from which it is reached.

Initialization: $S = U$ and $T = \varnothing$.

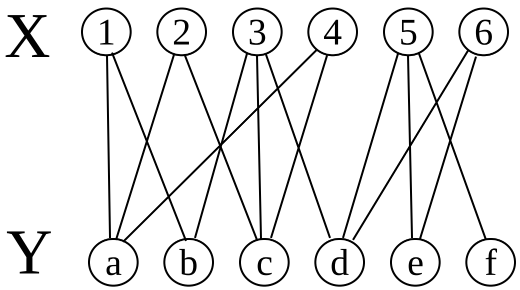
Algorithm: Augmenting Path Algorithm 3.2.1

Iteration:

- If S has no unmarked vertex, stop and report $T \cup (X - S)$ as a minimum cover and M as a maximum matching.
- Otherwise, select an unmarked $x \in S$. To explore x, consider each $y \in N(x)$ such that $xy \notin M$.
  - If y is unsaturated, terminate and report an M-augmenting path from U to y.
  - Otherwise, y is matched to some $w \in X$ by M. In this case, include y in T (reached from x) and include w in S (reached from y).
  - After exploring all such edges incident to x, mark x and iterate.

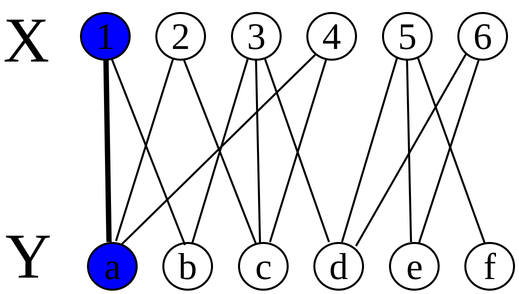


Example of Finding Matching$_1$



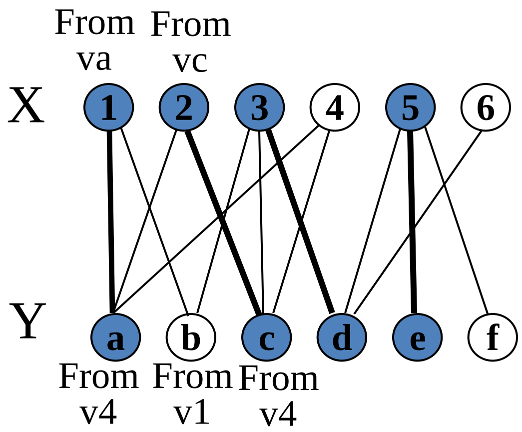
$M = \emptyset$, $U=\{1, 2, 3, 4, 5, 6\}$

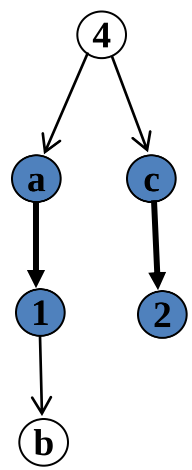
U: Unsaturated Vertices in X

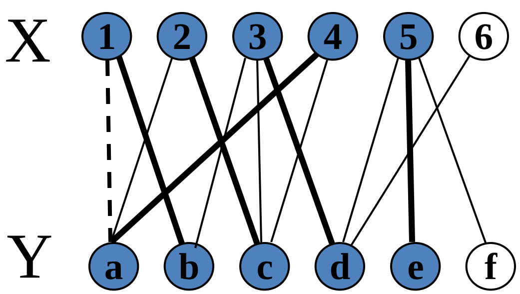


- Select one vertex from U, say $v1$.
- Consider the neighbors of $v1$ which is unsaturated.
- va is unsaturated.
- $v1$-va is an augmenting path.
- $M= \{(1,a)\}$, $U=\{2, 3, 4, 5, 6\}$

Example of Finding Matching[2]





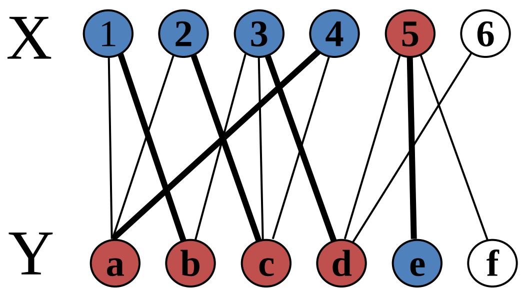


M = {(1,a)(2,c)(3,d)(5,e)}

U={4, 6}

- Select one vertex from U, say $v4$.
- Consider the neighbors of $v4$: va, vc
- Neither va nor vc is unsaturated
- Mark va and vc reached
- Consider the mate of va and the mate of vc
- Consider the neighbors of $v2$: va, vc
- Either va or vc is already reached
- Consider the neighbors of $v1$: va, vb
- vb is unsaturated,
- $v4$-va-$v1$-vb is an augmenting path.
- M= { (1,*b*)(2,*c*)(3,*d*)(4,*a*)(5,*e*)}
- U={6}

Example of Finding Matching[2]



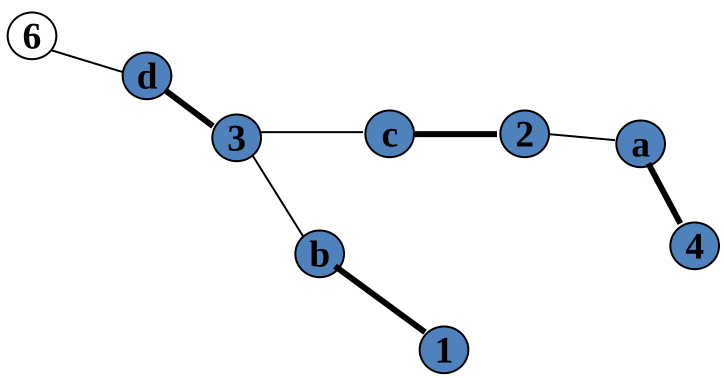
$M = \{(1,b)\ (2,c)\ (3,d)\ (4,a)\ (5,e)\}$

$U = \{6\}$

$S = \{1, 2, 3, 4, 6\}$

$T = \{a, b, c, d\}$

Vertex cover: $T \cup (X - S) = \{a, b, c, d, 5\}$



Theorem: Repeatedly applying the Augmenting Path Algorithm to a bipartite graph produces a matching and a vertex cover of equal size. 3.2.2

Proof: We need only verify that the Augmenting Path Algorithm produces an M-augmenting path or a vertex cover of size $|M|$.

- If the algorithm produces an M-augmenting path, we are finished.
- Otherwise, it terminates by marking all vertices of S and claiming that $Q = T \cup (X - S)$ is a vertex cover of size $|M|$.
- We must prove that Q is a vertex cover and has size $|M|$.

Theorem 3.2.2 Continue

- To show that Q is a vertex cover, it suffices to show that there is no edge joining S to Y - T.
  - An M-alternating path from U enters X only on an edge of M.
  - Hence every vertex x of S - U is matched via M to a vertex of T, and there is no edge of M from S to Y - T.
- Also there is no such edge outside M. When the path reaches $x \in S$, it can continue along any edge not in M, and exploring x puts all other neighbors of x into T.
- Since the algorithm marks all of S before terminating, all edges from S go to T.

Theorem 3.2.2 Continue

- Now we study the size of Q. The algorithm puts only saturated vertices in T; each $y \in T$ is matched via M to a vertex of S, Since $U \subseteq S$, also each vertex of $X - S$ is saturated, and the edges of M incident to $X - S$ cannot involve T.

- Hence they are different from the edges saturating T, and we find that M has at least $|T| + |X-S|$ edges. Since there is no matching large than this vertex cover, we have $|M|=|T| + |X-S| = |Q|$

Remark 3.2.4

- Let G be an X,Y-bigraph with n vertices and m edges. Since $\alpha'(G) \le n/2$ we find a maximum matching in G by applying **Algorithm 3.2.1 (Augmenting Path Algorithm)** at most $n/2$ times.

- Each application explores a vertex of X at most once, just before marking it; thus it considers each edge at most once. If the time for one edge exploration is bounded by a constant, then this algorithm to find a maximum matching runs in time ***O*(*nm*)**.

- The **Hopcroft-Karp [1973] algorithm** is a faster algorithm, with running time $O(\sqrt{n}m)$

Conclusion

- In this lecture, we have discussed König-Egerváry theorem, Independent sets, Covers *i.e.* edge cover, vertex cover, Maximum bipartite matching and Augmenting Path Algorithm.

- In upcoming lectures, we will discuss weighted bipartite matching, stable matching and faster bipartite matching.

Weighted Bipartite Matching

Dr. Rajiv Misra
Associate Professor
Dept. of Computer Science & Engg.
Indian Institute of Technology Patna
rajivm@iitp.ac.in

Preface

Recap of Previous Lecture:

- In the previous lecture, we have discussed the Concept of Matching, Perfect matchings, Maximal matchings, Maximum Matchings, M-alternating path, M-augmenting path, Symmetric difference, Hall's Matching condition and Vertex covers.

Content of this Lecture:

- In this lecture, we will discuss König-Egerváry theorem, Independent sets, Covers i.e. edge cover, vertex cover, Maximum bipartite matching, Augmenting Path Algorithm and Running time.

Weighted Bipartite Matching

- Seek a matching of maximum total weight.
- It is assumed that the given graph is $K_{n,n}$.
  - If the given graph is not a complete bipartite graph, insert edges with zero weight.

Examples of Weighted bipartite matching and its dual 3.2.5

- A farming company owns n farms and n processing plants.
  - Each farm can produce corn to the capacity of one plant.
  - The profit that results from sending the output of farm i to plant j is $w_{i,j}$.
  - Placing weight $w_{i,j}$ on edge $x_i y_j$ gives us a weighted bipartite graph with partite sets $X=\{x_1,\ldots,x_n\}$ and $Y=\{y_1,\ldots,y_n\}$.
  - The company wants to select edges forming a matching to maximize total profit.

Examples of Weighted bipartite matching and its dual

- The government claims that too much corn is being produced, so it will pay the company not to process corn.
  - The government will pay u_i if the company agrees not to use farm i and v_j if it agrees not to use plant j.
  - If $u_i+v_j<w_{i,j}$, then the company makes more by using the edge x_iy_j than by taking the government payments for those vertices.
  - In order to stop all production, the government must offer amounts such that $u_i+v_j \geq w_{i,j}$ for all i, j. The governments wants to find such values to minimize $\Sigma u_i + \Sigma v_j$.

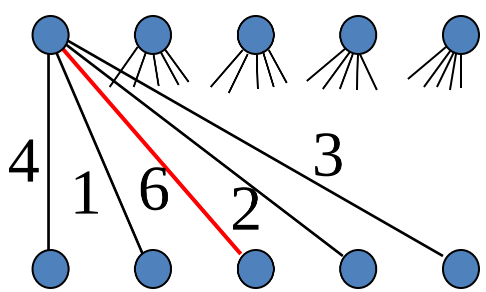
Transversal of an *n*-by-*n* matrix

- A ***transversal*** of an *n*-by-*n* matrix consists of *n* positions, one in each row and each column.
- Finding a transversal with maximum sum is the **Assignment Problem**.

$$\begin{pmatrix} 4 & 1 & 6 & 2 & 3 \\ 5 & 0 & 3 & 7 & 6 \\ 2 & 3 & 4 & 5 & 8 \\ 3 & 4 & 6 & 3 & 4 \\ 4 & 6 & 5 & 8 & 6 \end{pmatrix}$$

Assignment Problem

- This is the matrix formulation of the **maximum weighted matching** problem, where nonnegative weight $w_{i,j}$ is assigned to edge x_iy_j of $K_{n,n}$ and
- We seek a perfect matching M to maximize the total weight $w(M)$.



$$\begin{pmatrix} 4 & 1 & 6 & 2 & 3 \\ 5 & 0 & 3 & 7 & 6 \\ 2 & 3 & 4 & 5 & 8 \\ 3 & 4 & 6 & 3 & 4 \\ 4 & 6 & 5 & 8 & 6 \end{pmatrix}$$

Minimum weighted Cover

- With these weights, a **(weighted) cover** is a choice of labels $u_i,\ldots,u_n$ and $v_j,\ldots,v_n$ such that $u_i+v_j \geq w_{i,j}$ for all i, j. The **cost** $c(u, v)$ for a cover (u, v) is $\Sigma u_i+\Sigma v_j$.
- The **minimum weighted cover** problem is that of finding a cover of minimum cost.

$$\begin{array}{c c}
 & \begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \end{array} \\
\begin{array}{c} 6 \\ 7 \\ 8 \\ 6 \\ 8 \end{array} & \begin{pmatrix} 4 & 1 & \underline{6} & 2 & 3 \\ 5 & 0 & 3 & \underline{7} & 6 \\ 2 & 3 & 4 & 5 & \underline{8} \\ 3 & 4 & \underline{6} & 3 & 4 \\ 4 & 6 & 5 & \underline{8} & 6 \end{pmatrix}
\end{array}$$

Lemma: For a perfect matching M and *cover*(u, v) in a weighted bipartite graph G, also $c(u, v) = w(M)$ if and only if M consists of edges $x_i y_i$ such that $u_i + v_j = w_{i,j}$. In this case, M and (u,v) are optimal. 3.2.7

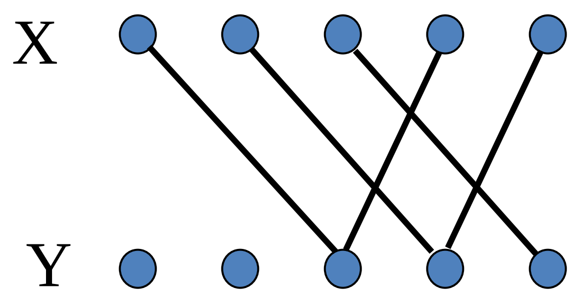
Proof:

- Since M saturates each vertex, summing the constraints $u_i + v_j \geq w_{i,j}$ that arise from its edges yields $c(u, v) = w(M)$, then equality must hold in each of the n inequalities summed.
- Finally, since $c(u, v) \geq w(M)$ for every matching and every cover, $c(u, v) = w(M)$ implies that there is no matching with weight greater than $c(u, v)$ and no cover with cost less than $w(M)$.

Equality subgraph 3.2.8

- The *equality subgraph* $G_{u,v}$ for a cover (u, v) is the spanning subgraph of $K_{n,n}$ having the edges x_iy_j such that $u_i+v_j=w_{i,j}$.

$$\begin{array}{ccc} & & Y \\ & & \begin{matrix} 0 & 0 & 0 & 0 & 0 \end{matrix} \\ X & \begin{matrix} 6 \\ 7 \\ 8 \\ 6 \\ 8 \end{matrix} & \begin{pmatrix} 4 & 1 & \underline{6} & 2 & 3 \\ 5 & 0 & 3 & \underline{7} & 6 \\ 2 & 3 & 4 & 5 & \underline{8} \\ 3 & 4 & \underline{6} & 3 & 4 \\ 4 & 6 & 5 & \underline{8} & 6 \end{pmatrix} \end{array}$$

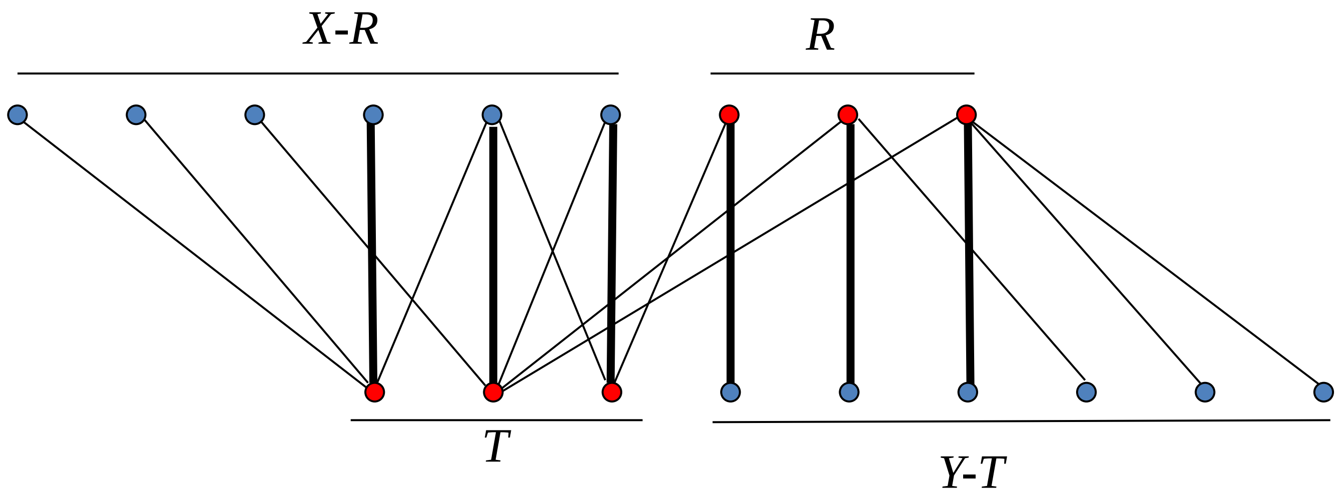


Equality Subgraph 3.2.8

- If $G_{u,v}$ has a perfect matching, then its weight is $\Sigma u_i + \Sigma v_j$, and by Lemma 3.2.7 we have the optimal solution.

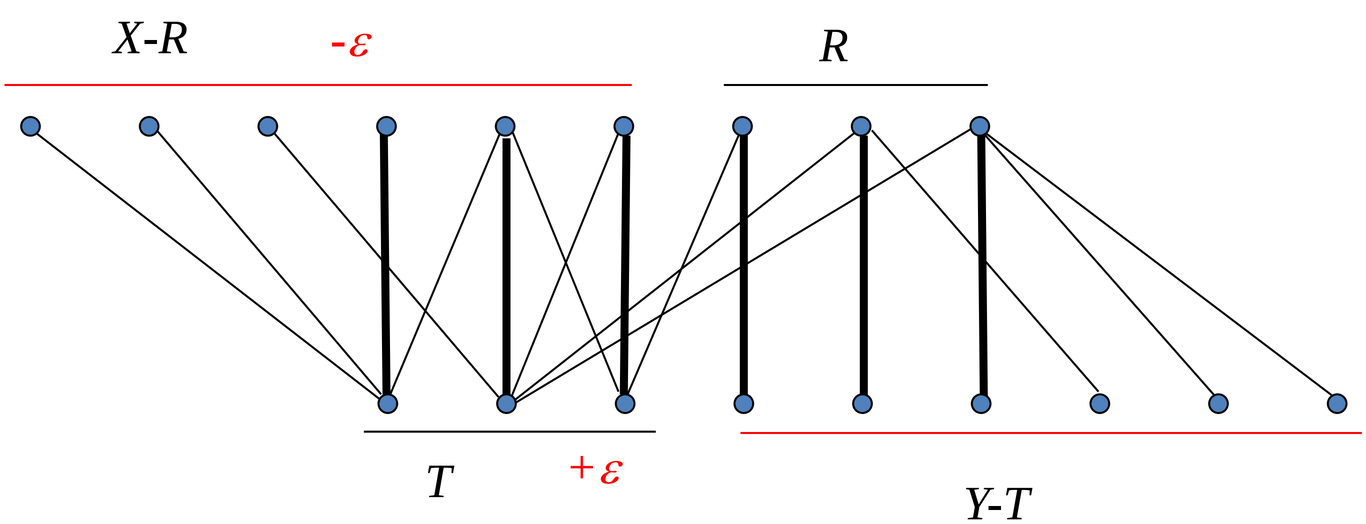
Equality Subgraph Continue

- Otherwise, we find a matching M and a vertex cover Q of the same size in $G_{u,v}$ (by using the Augmenting Path Algorithm, for example). Let $R = Q \cap X$ and $T = Q \cap Y$. Our matching of size $|Q|$ consists of $|R|$ edges from R to $Y-T$ and $|T|$ edges from T to $X-R$, as shown below.



Equality Subgraph Continue

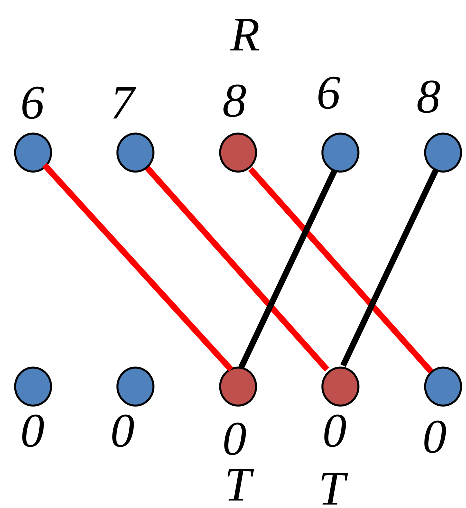
- Enlarge the equality subgraph so that there is a larger matching in the new equality subgraph,
- We change (u, v) to introduce an edge from X-R to Y-T while maintaining equality on all edges of M.



Equality Subgraph Continue

- A cover requires $u_i+v_j \geq w_{i,j}$ for all i, j; the difference $u_i+v_j-w_{i,j}$ is the **excess** for i, j.
- Edges joining X-R and Y-T are not in $G_{u,v}$ and have positive excess.

$$\begin{array}{cc|ccccc}
 & & & & T & T & \\
 & & 0 & 0 & 0 & 0 & 0 \\
\hline
 & 6 & 4 & 1 & \underline{6} & 2 & 3 \\
 & 7 & 5 & 0 & 3 & \underline{7} & 6 \\
R & 8 & 2 & 3 & 4 & 5 & \underline{8} \\
 & 6 & 3 & 4 & \underline{6} & 3 & 4 \\
 & 8 & 4 & 6 & 5 & \underline{8} & 6
\end{array}$$



R∪T: Vertex Cover

Equality Subgraph Continue

- Let ε be the minimum excess on the edges from *X*-*R* to *Y*-*T*. Reduce u_i by ε for all $x_i \in X\text{-}R$
- To maintain the cover condition for these edges while bringing at least one into the equality subgraph
  - Increase v_j by ε for $y_j \in T$
  - To maintain the cover condition for the edges from *X*-*R* to *T*

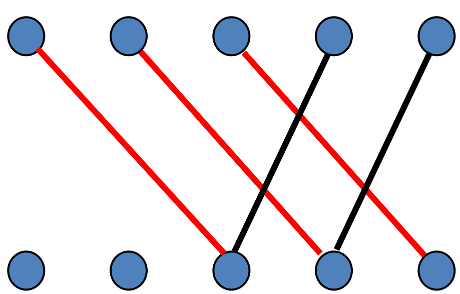
$$
\begin{array}{ccc}
 & & \begin{array}{ccccc} & & T & T & \\ 0 & 0 & 0 & 0 & 0 \end{array} \\
\begin{array}{c} \\ \\ R \\ \\ \\ \end{array} & \begin{array}{c} 6 \\ 7 \\ 8 \\ 6 \\ 8 \end{array} & \begin{pmatrix} 4 & 1 & \underline{6} & 2 & 3 \\ 5 & 0 & 3 & \underline{7} & 6 \\ 2 & 3 & 4 & 5 & \underline{8} \\ 3 & 4 & \underline{6} & 3 & 4 \\ 4 & 6 & 5 & \underline{8} & 6 \end{pmatrix}
\end{array}
\qquad
\begin{array}{ccc}
 & & \begin{array}{ccccc} & & T & T & \\ 0 & 0 & 0 & 0 & 0 \end{array} \\
\begin{array}{c} \\ \\ R \\ \\ \\ \end{array} & \begin{array}{c} 6 \\ 7 \\ 8 \\ 6 \\ 8 \end{array} & \begin{pmatrix} 2 & 5 & \underline{0} & 4 & 3 \\ 2 & 7 & 4 & \underline{0} & 1 \\ 6 & 5 & 4 & 3 & \underline{0} \\ 3 & 2 & \underline{0} & 3 & 2 \\ 4 & 2 & 3 & \underline{0} & 2 \end{pmatrix}
\end{array}
$$

Matrix of excess

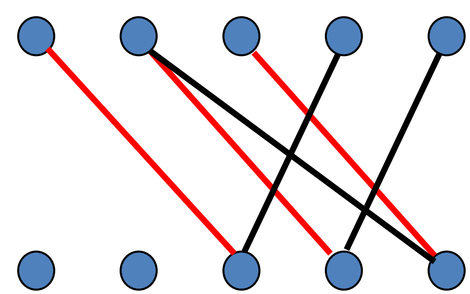
Min excess ε = 1

Equality Subgraph Continue

$$
\begin{array}{cc|c}
 & & \begin{array}{ccccc} & & T & T & \\ 0 & 0 & 0 & 0 & 0 \end{array} \\
 & \begin{array}{c} 6 \\ 7 \\ 8 \\ 6 \\ 8 \end{array} & \begin{pmatrix} 4 & 1 & \underline{6} & 2 & 3 \\ 5 & 0 & 3 & \underline{7} & 6 \\ 2 & 3 & 4 & 5 & \underline{8} \\ 3 & 4 & \underline{6} & 3 & 4 \\ 4 & 6 & 5 & \underline{8} & 6 \end{pmatrix} \\
\end{array}
\quad R \text{ (row labels)}
$$



$$
\begin{array}{cc|c}
 & & \begin{array}{ccccc} & & T & T & \\ 0 & 0 & \boxed{1} & \boxed{1} & 0 \end{array} \\
R & \begin{array}{c} \boxed{5} \\ \boxed{6} \\ 8 \\ \boxed{5} \\ \boxed{7} \end{array} & \begin{pmatrix} 4 & 1 & \underline{6} & 2 & 3 \\ 5 & 0 & 3 & \underline{7} & \underline{6} \\ 2 & 3 & 4 & 5 & \underline{8} \\ 3 & 4 & \underline{6} & 3 & 4 \\ 4 & 6 & 5 & \underline{8} & 6 \end{pmatrix} \\
\end{array}
$$



Equality subgraph is expanded

Equality Subgraph Continue

- Repeat the procedure with the new equality subgraph; eventually we obtain a cover whose equality subgraph has a perfect matching.

Hungarian Algorithm 3.2.9

Input: A matrix of weights on the edges of $K_{n,n}$ with bipartition X,Y.

Idea: Iteratively adjusting the cover (u, v) until the equality subgraph $G_{u,v}$ has a perfect matching.

Initialization: Let (u, v) be a cover, such as $u_i = \max_j w_{i,j}$ and $v_j = 0$.

Hungarian Algorithm Continue

Iteration: Find a maximum matching M in $G_{u,v}$.

- If M is a perfect matching, stop and report M as a maximum weight matching.
- Otherwise,
  - Let Q be a vertex cover of size $|M|$ in $G_{u,v}$.
  - Let $R=X\cap Q$ and $T=Y\cap Q$.
  - Let $\varepsilon=\min\{u_i+v_j-w_{i,j}: x_i\in X-R, y_j\in Y-T\}$.
  - Decrease u_i by ε for $x_i\in X-R$, and
  - increase v_j by ε for $y_j\in T$.
- Form the new equality subgraph and repeat.

Solving the Assignment Problem 3.2.10

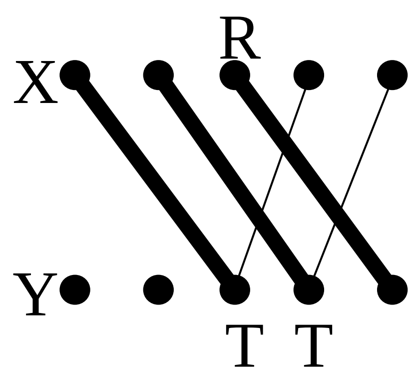
- The first matrix below is the matrix of weights.
- The others display a cover (u,v) and the corresponding excess matrix.

$$
\begin{pmatrix} 4 & 1 & 6 & 2 & 3 \\ 5 & 0 & 3 & 7 & 6 \\ 2 & 3 & 4 & 5 & 8 \\ 3 & 4 & 6 & 3 & 4 \\ 4 & 6 & 5 & 8 & 6 \end{pmatrix}
\Rightarrow
\begin{array}{c} \\ 6 \\ 7 \\ 8 \\ 6 \\ 8 \\ \\ \end{array}
\begin{array}{c} 0 \quad 0 \quad 0 \quad 0 \quad 0 \\ \begin{pmatrix} 2 & 5 & \underline{0} & 4 & 3 \\ 2 & 7 & 4 & \underline{0} & 1 \\ 6 & 5 & 4 & 3 & \underline{0} \\ 3 & 2 & 0 & 3 & 2 \\ 4 & 2 & 3 & 0 & 2 \end{pmatrix} \\ \quad \quad T \quad T \quad \end{array}
\begin{array}{c} \\ \\ \\ R \\ \\ \\ \\ \end{array}
$$

Solving the Assignment Problem 3.2.10

- We underscore entries in the excess matrix to mark a maximum matching M of $G_{u,v}$, which appears as bold edges in the equality subgraphs drawn for the first two excess matrices. (Drawing the equality subgraphs is not necessary.)
- A matching in $G_{u,v}$ corresponds to a set of 0s in the excess matrix with no two in any row or column; call this a **partial transversal**.

$$\begin{array}{ccc} & \begin{array}{ccccc}0&0&0&0&0\end{array} & \\ \begin{array}{c}6\\7\\8\\6\\8\end{array} & \begin{pmatrix} 2&5&\underline{0}&4&3 \\ 2&7&4&\underline{0}&1 \\ 6&5&4&3&\underline{0} \\ 3&2&0&3&2 \\ 4&2&3&0&2 \end{pmatrix} & \begin{array}{c} \\ \\ R \\ \\ \\ \end{array} \\ & \begin{array}{ccccc} & & T & T & \end{array} & \end{array}$$

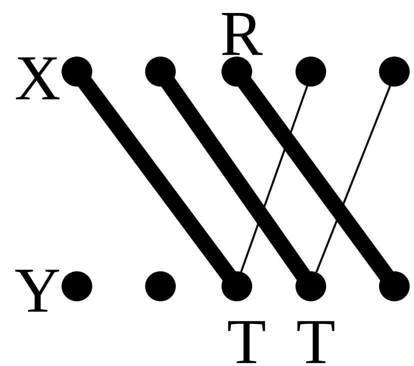


Solving the Assignment Problem

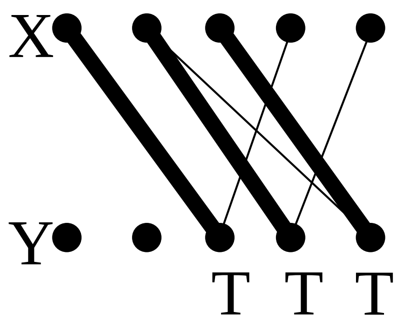
- A set of rows and columns covering the 0s in the excess matrix is a **covering set**; this corresponds to a vertex cover in $G_{u,v}$. A covering set of size less than n yields progress toward a solution, since the next weighted cover costs less. We study the 0s in the excess matrix and find a partial transversal and a covering set of the same size. In a small matrix, we can do this by inspection.

Solving the Assignment Problem

$$\begin{pmatrix} 4 & 1 & 6 & 2 & 3 \\ 5 & 0 & 3 & 7 & 6 \\ 2 & 3 & 4 & 5 & 8 \\ 3 & 4 & 6 & 3 & 4 \\ 4 & 6 & 5 & 8 & 6 \end{pmatrix} \Rightarrow \begin{array}{c} \\ 6 \\ 7 \\ 8 \\ 6 \\ 8 \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \end{array} \\ \begin{pmatrix} 2 & 5 & \underline{0} & 4 & 3 \\ 2 & 7 & 4 & \underline{0} & 1 \\ 6 & 5 & 4 & 3 & \underline{0} \\ 3 & 2 & 0 & 3 & 2 \\ 4 & 2 & 3 & 0 & 2 \end{pmatrix} \\ \begin{array}{ccccc} & & T & T & \end{array} \end{array} \begin{array}{c} \\ \\ \\ R \\ \\ \\ \\ \end{array}$$



$$\begin{array}{c} \\ 5 \\ 6 \\ 8 \\ 5 \\ 7 \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 0 & 1 & 1 & 0 \end{array} \\ \begin{pmatrix} 1 & 4 & \underline{0} & 4 & 2 \\ 1 & 6 & 4 & \underline{0} & 0 \\ 6 & 5 & 5 & 4 & \underline{0} \\ 2 & 1 & 0 & 3 & 1 \\ 3 & 1 & 3 & 0 & 1 \end{pmatrix} \\ \begin{array}{ccccc} & & T & T & T \end{array} \end{array}$$



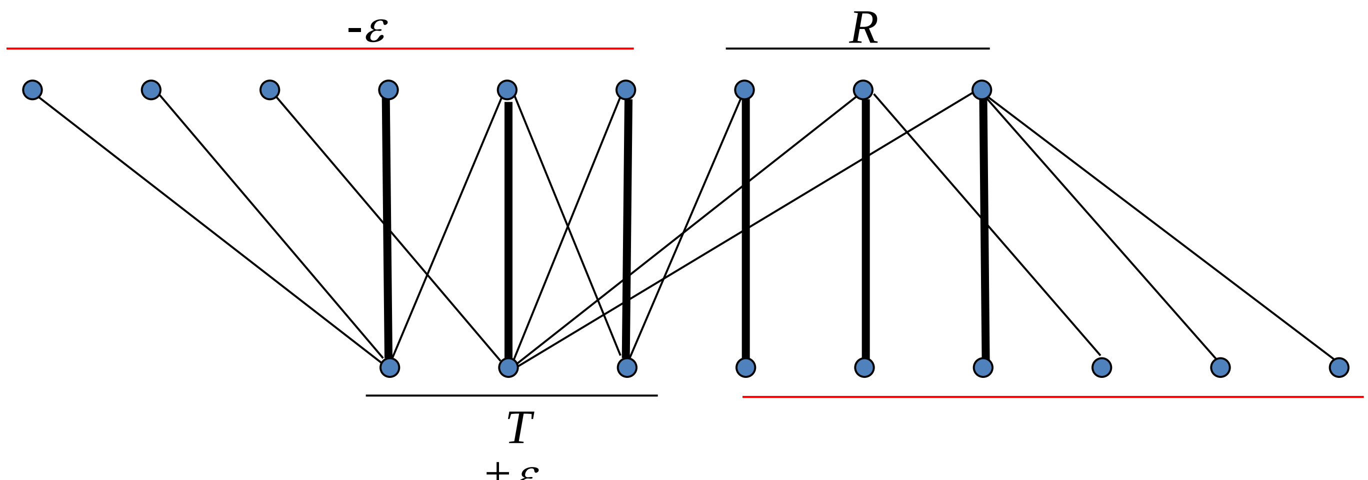
$$\Rightarrow \begin{array}{c} \\ 4 \\ 5 \\ 7 \\ 4 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 0 & 2 & 2 & 1 \end{array} \\ \begin{pmatrix} 0 & 3 & \underline{0} & 4 & 2 \\ \underline{0} & 5 & 4 & 0 & 0 \\ 5 & 4 & 5 & 4 & \underline{0} \\ 1 & \underline{0} & 0 & 3 & 1 \\ 2 & 0 & 3 & \underline{0} & 1 \end{pmatrix} \end{array}$$

Theorem: The hungarian Algorithm finds a maximum weight matching and a minimum cost cover. 3.2.11

- The algorithm begins with a cover. It can terminate only when the equality subgraph has a perfect matching, which guarantees equal value for the current matching and cover.
- Suppose that (u, v) is the current cover and that the equality subgraph has no perfect matching.
- Let (u', v') denote the new lists of numbers assigned to the vertices. Because ε is the minimum of a nonempty finite set of positive numbers, $\varepsilon>0$.

Theorem 3.2.11 Continue

- We verify first that (u', v') is a cover.
  - The change of labels on vertices of X-R and T yields $u_i'+v_j'=u_i+v_j$ for edges x_iy_j from X-R to T or from R to Y-T.
  - If $x_i \in R$ and $y_i \in T$, then $u_i'+v_j'=u_i+v_j+\varepsilon$, and the weight remains covered.
  - If $x_i \in X$-R and $y_j \in Y$-T, then $u_i'+v_j'$ equals $u_i+v_j-\varepsilon$, which by the choice of ε is at least $w_{i,j}$.



Theorem 3.2.11 Continue

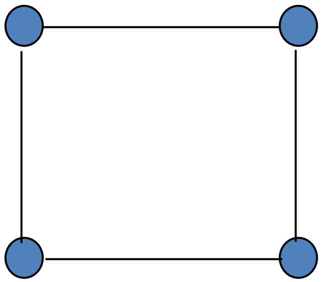
- The algorithm terminates only when the equality subgraph has a perfect matching, so it suffices to show that it does terminate.
- Suppose that the weights $w_{i,j}$ are rational. Multiplying the weights by their least common denominator yields an equivalent problem with integer weights.

Theorem 3.2.11 Continue

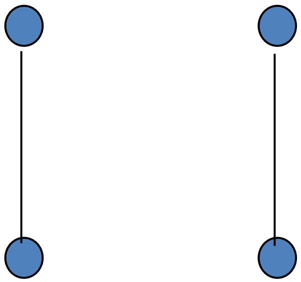
- We can now assume that the labels in the current cover also are integers.
- Thus each excess is also an integer, and at each iteration we reduce the cost of the cover by an integer amount.
- Since the cost starts at some value and is bounded bellow by the weight of a perfect matching, after finitely many iterations we have equality.

Factor

- A *factor* of graph *G* is a spanning subgraph of *G*.
  - A *k-factor* is a spanning *k*-regular subgraph.
  - An odd component of a graph is a component of odd order; the number of odd components of *H* is *o*(*H*).
- A 1-factor and a perfect matching are almost the same thing.



G



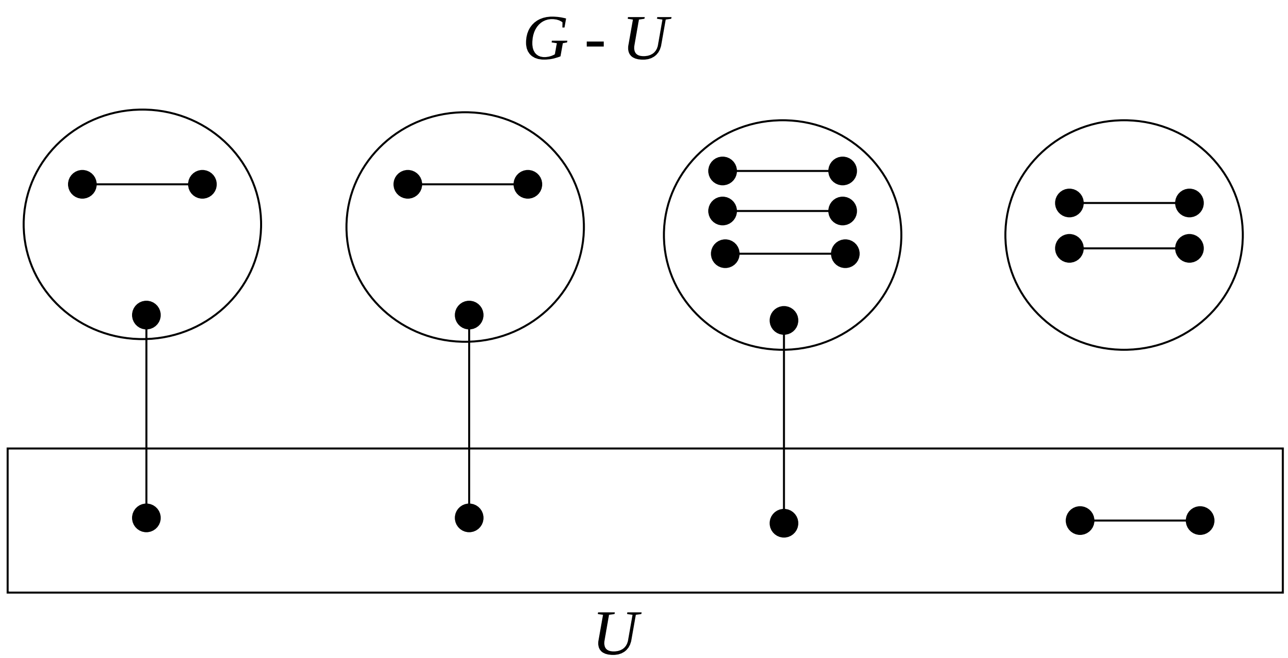
1-factor

Tutte's 1-factor Theorem

- Tutte found a necessary and sufficient condition for which graphs have 1-factors.
  - If G has a 1-factor and we consider a set $S \subseteq V(G)$, then every odd component of G-S has a vertex matched to something outside it, which can only belong to S.
  - Since these vertices of S must be distinct, $o(G\text{-}S) \leq |S|$.
- **Tutte's Condition:** For all $S \subseteq V(G)$, $o(G\text{-}S) \leq |S|$.
  - Tutte proved that this obvious necessary condition is also sufficient (TONCAS).

Theorem: A graph G has a 1-factor if and only if $o(G\text{-}S) \leq |S|$ for every $S \subseteq V(G)$ 3.3.3

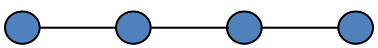
- Example:



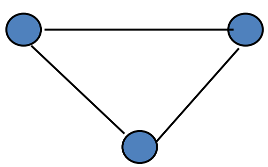
Join 3.3.6

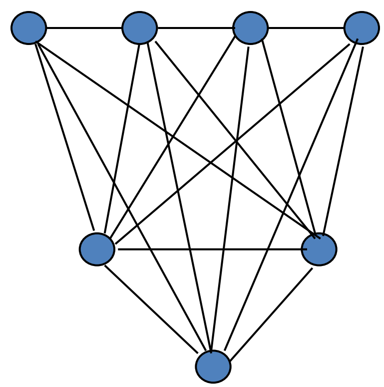
- The join of simple graphs G and H, written $G \vee H$, is the graph obtained from the disjoint union $G+H$ by adding the edges $\{xy : x \in V(G),\ y \in V(H)\}$.

P4



K3





P4 $\vee$ K3

Corollary: The largest number of vertices saturated by a matching in G is $\min_{S \subseteq V(G)} \{n(G)-d(S)\}$, where $d(S) = o(G-S) - |S|$. 3.3.7

Proof:

- Given $S \subseteq V(G)$, at most $|S|$ edges can match vertices of S to vertices in odd components of G-S, so every matching have at least $o(G-S)-|S|$ unsaturated vertices. We want to achieve this bound.
- Let $d=\max\{o(G-S)-|S|: S \subseteq V(G)\}$. The case $S=\varnothing$ yields $d \geq 0$. Let $G' = G \vee K_d$. Since $d(S)$ has the same parity as $n(G)$ for each S, we know that $n(G')$ is even. If G' satisfies Tutte's Condition, then we obtain a matching of the desired size in G from a perfect matching in G', because deleting the d added vertices eliminates edges that saturate at most d vertices of G.

Corollary 3.3.7 continue

- The condition $o(G'-S')\leq|S'|$ holds for $S'=\varnothing$ because $n(G')$ is even.
- If S' is nonempty but does not contain all of K_d, then $G'-S'$ has only one component, and $1<|S'|$.
- Finally, when $K_d\subseteq S'$, we let $S=S'-V(K_d)$. We have $G'-S'=G-S$, so $o(G'-S')=o(G-S)\leq|S|+d=|S'|$.
- We have verified that G' satisfies Tutte's Condition

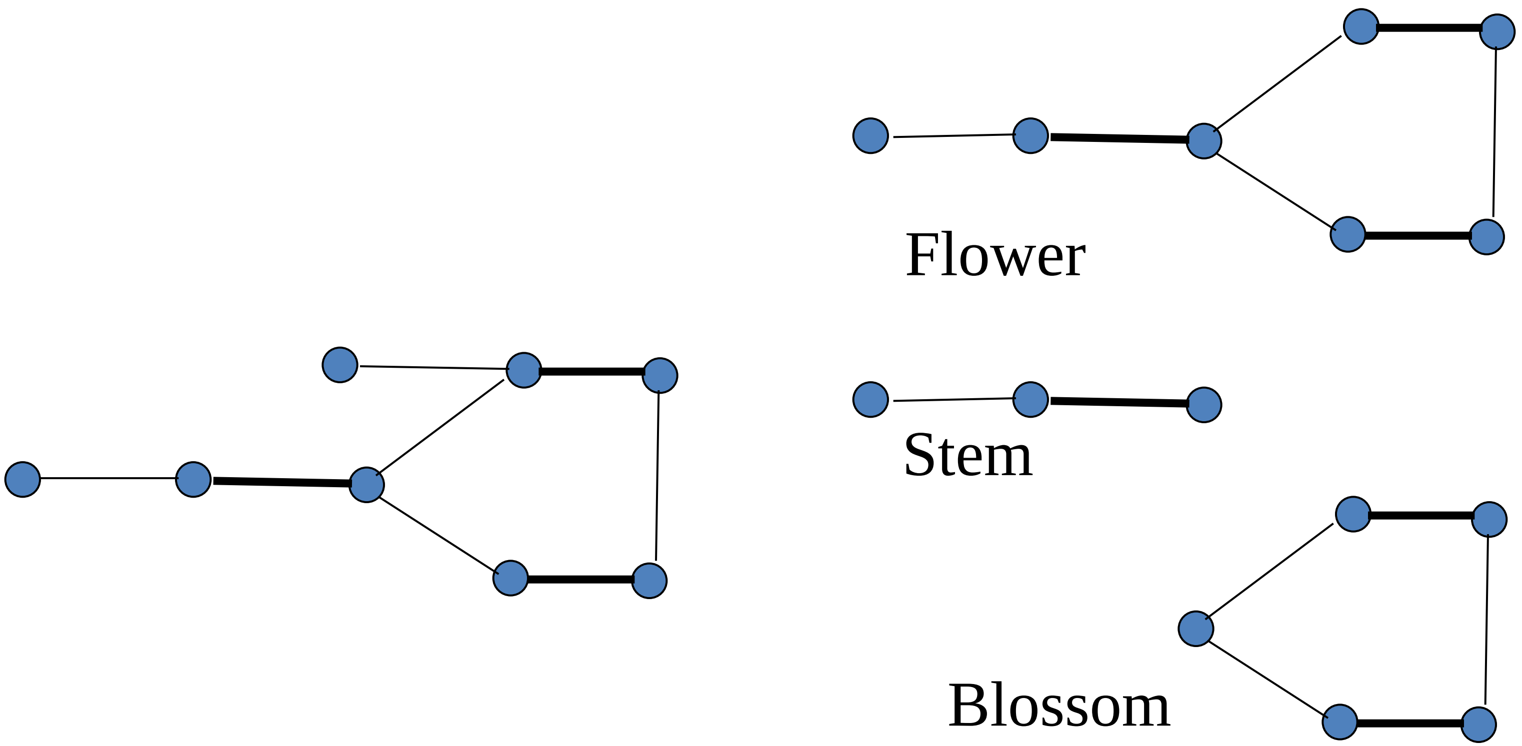
Edmonds' Blossom Algorithm

- In bipartite graphs, we can search quickly for augmenting paths (Algorithm 3.2.1) because we explore from each vertex at most once.
- An M-alternating path from u can reach a vertex x in the same partite set as u only along a saturated edge. Hence only once can we search and explore x.
- This property fails in graphs with odd cycles, because M-alternating paths from an unsaturated vertex may reach x both along saturated and along unsaturated edges.

Flower, Stem, Blossom

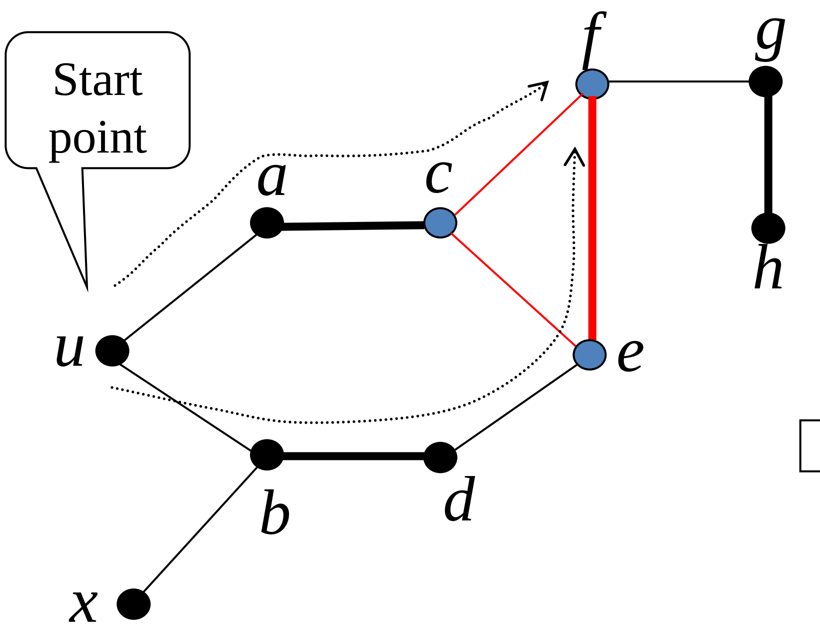
- That M be a matching in a graph G, and let *u* be an M-unsaturated vertex.
  - A **flower** is the union of two M-alternating paths from *u* that reach a vertex *x* on steps of opposite parity (having not done so earlier).
  - The **stem** of the flower is the maximal common initial path (of nonnegative even length).
  - The **blossom** of the flower is the odd cycle obtained by deleting the stem.

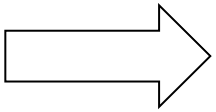
Example of Flower, Stem, and Blossom



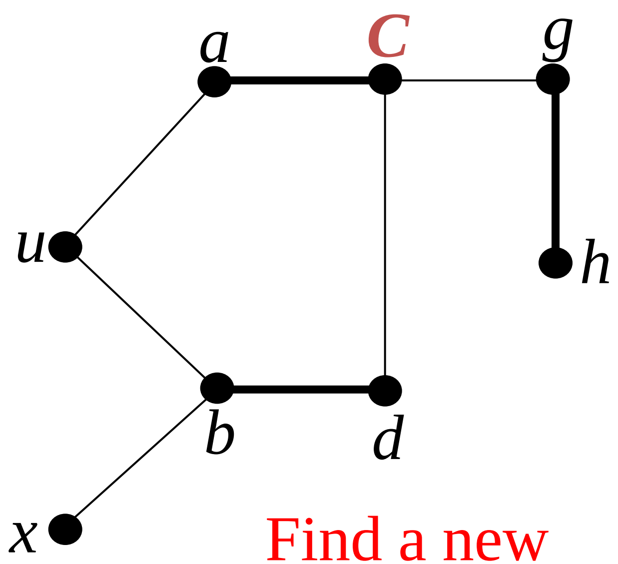
Example 3.3.16

Blossom



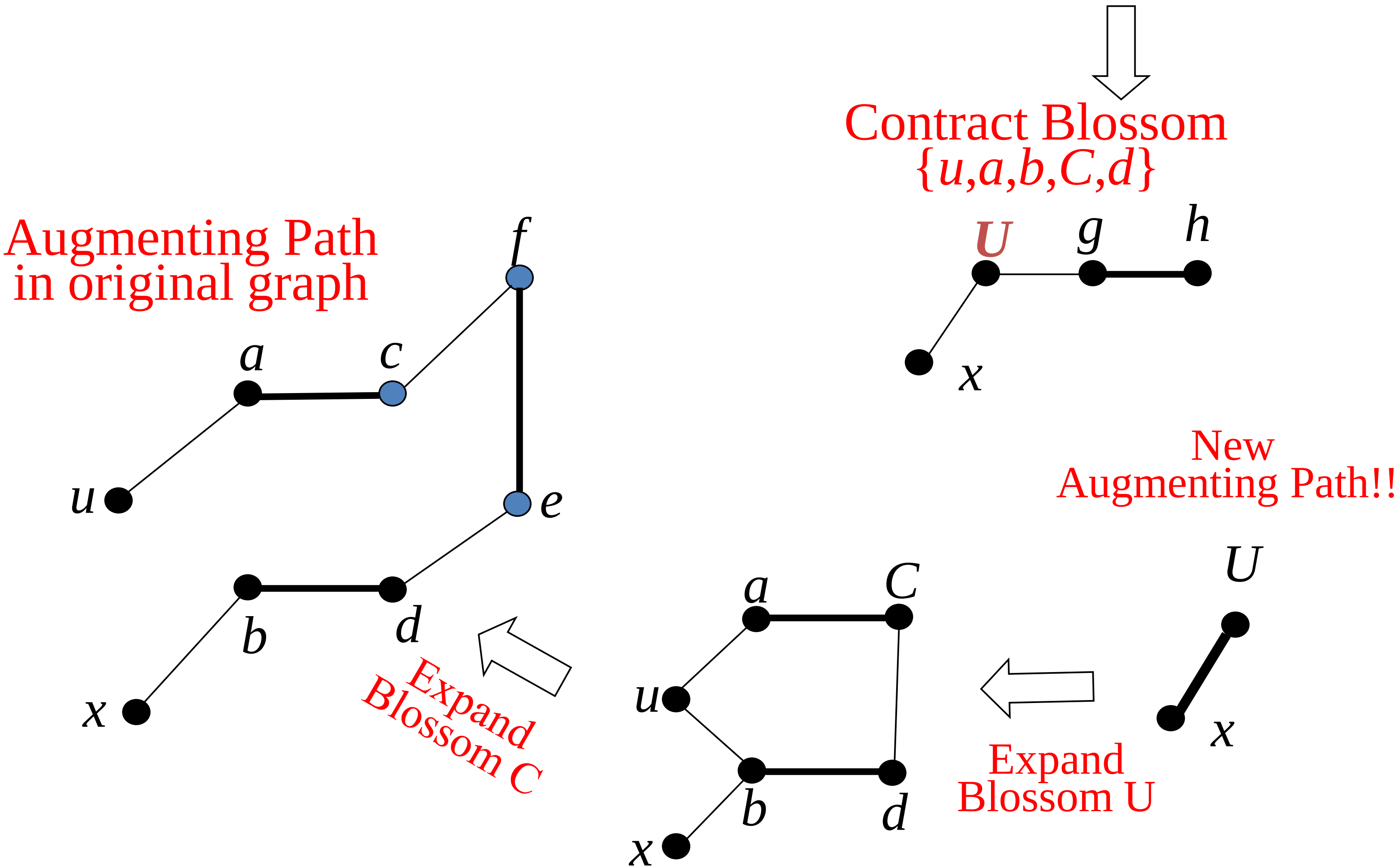


Contract Blossom
$\{c, e, f\}$

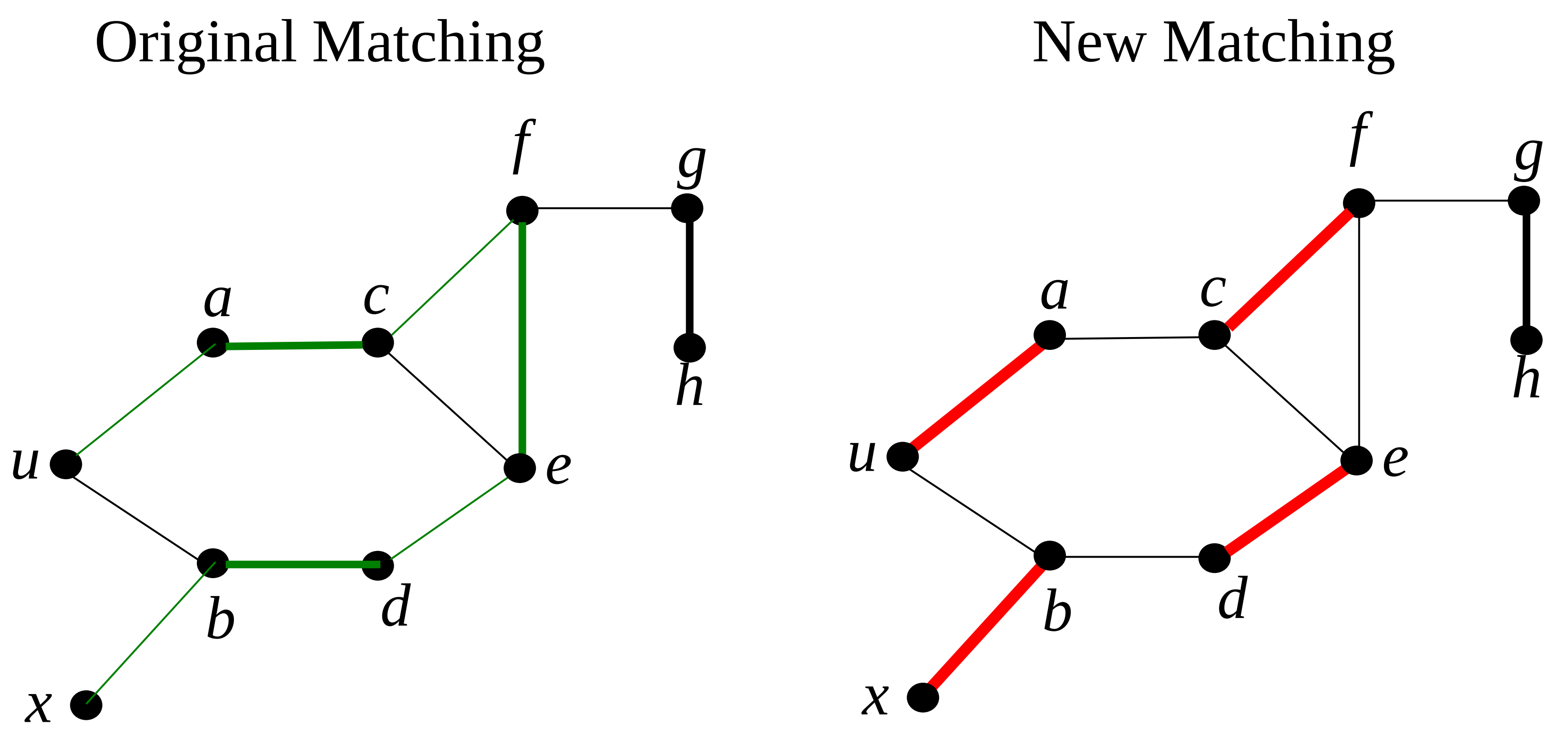


Find a new Blossom
$\{u,a,b,C,d\}$

Example


Contract Blossom
{u,a,b,C,d}
U
g
h
x
Augmenting Path
in original graph
f
a
c
u
e
b
d
x
Expand
Blossom C
New
Augmenting Path!!!
a
C
u
b
d
x
U
x
Expand
Blossom U

Example 3.3.16



Edmonds' Blossom Algorithm$_{3.3.17}$

Input: A graph G, a matching M in G, an M-unsaturated vertex u.

Idea: Explore M-altenating paths from u, recording for each vertex the vertex from which it was reached, and contracting blossoms when found. Maintain sets S and T analogous to those in Algorithm 3.2.1, with S consisting of u and the vertices reached along saturated edges. Reaching an unsaturated vertex yields an augmentation.

Initialization: $S=\{u\}$and $T=\varnothing$

Edmonds' Blossom Algorithm 3.3.17

Iteration:

- If S has no unmarked vertex, stop; there is no M-augmenting path from u. Otherwise, select an unmarked $v \in S$. To explore from v, successively consider each $y \in N(v)$ such that $y \notin T$.
- If y is unsaturdated by M, then trace back from y (expanding blossoms as needed) to report an M-augmrnting u,y-path.
- If $y \in S$, then a blossom has been found. Suspend the exploration of v and contract the blossom, replacing its vertices in S and T by a single new vertex in S. Continue the search from this vertex in the smaller graph. Otherwise, y is matched to some w by M. Include y in T (reached from v), and include w in S (reached from y).
- After exploring all such neighbors of v, mark v and iterate.