Lecture: Matchings (3)

Shuai Li

John Hopcroft Center, Shanghai Jiao Tong University

https://shuaili8.github.io

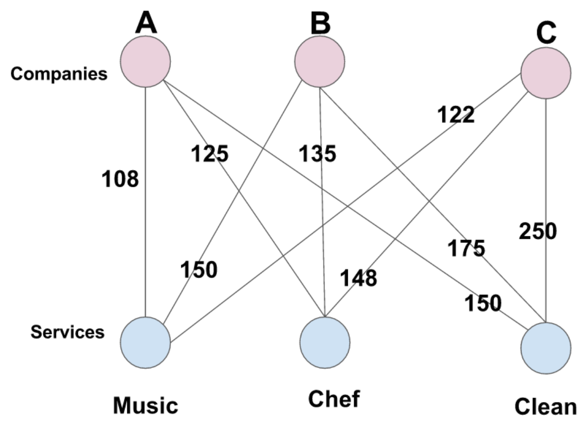
https://shuaili8.github.io/Teaching/CS445/index.html

Hungarian algorithm

- **Input**: Weighted $K_{n,n} = B(X, Y)$
- **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 $u_i = \max_j w_{i,j}$, $v_j = 0$

(Dual)

$$\min \sum_i u_i + \sum_j v_j$$
$$s.t. \, u_i + v_j \geq w_{i,j} \text{ for any } i, j$$
$$u_i, v_j \geq 0$$

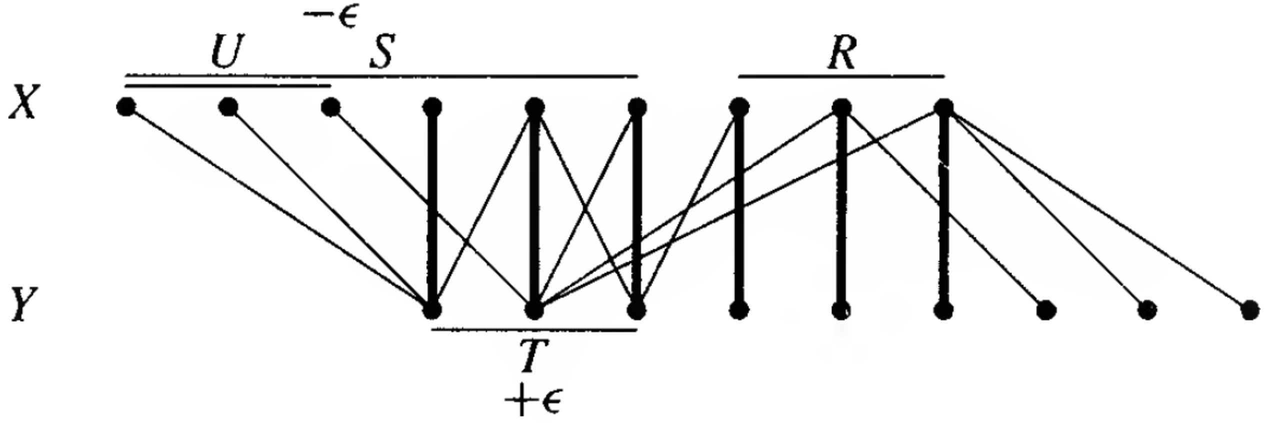


Hungarian algorithm (cont.)

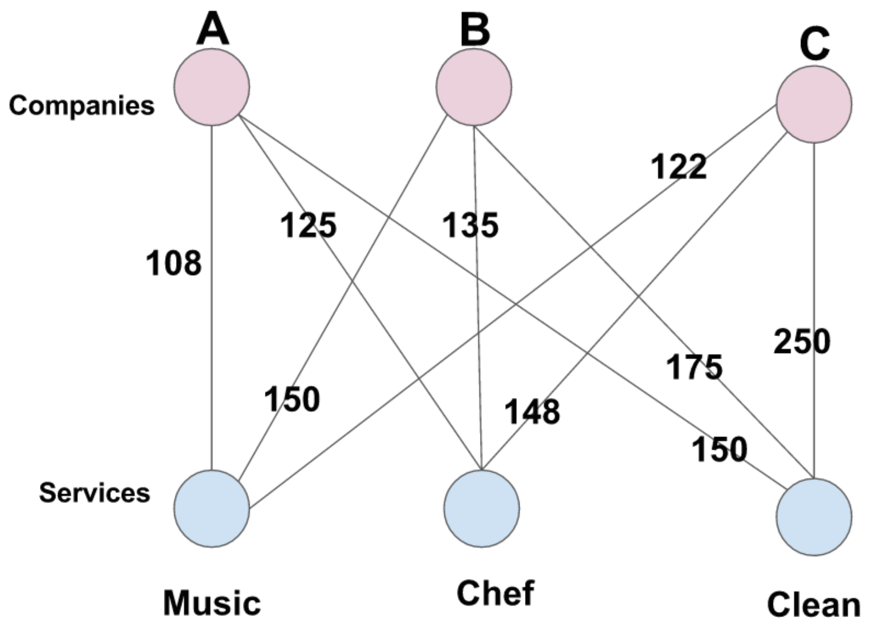
- **Iteration**: Find a maximum matching M in $G_{u,v}$
  - If M is a perfect matching, stop and report M as a maximum weight matching
  - Otherwise, let Q be a vertex cover of size $|M|$ in $G_{u,v}$
    - Let $R = X \cap Q, T = Y \cap Q$

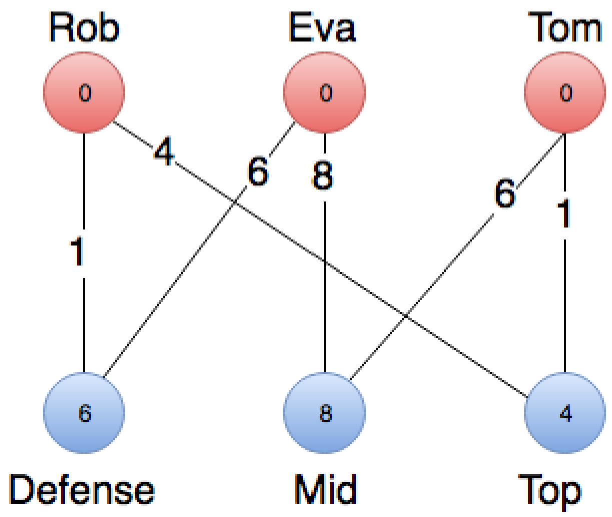
$$\epsilon = \min\{u_i + v_j - w_{i,j} : x_i \in X - R, y_j \in Y - T\}$$

    - 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



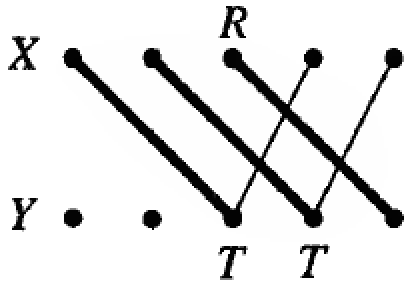
Example


A
B
C
Companies
108
125
135
122
250
175
150
148
150
Services
Music
Chef
Clean

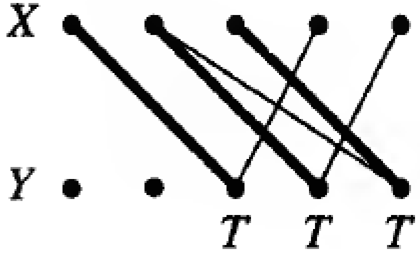

Rob
Eva
Tom
0
0
0
4
6
8
6
1
1
6
8
4
Defense
Mid
Top

Example 2

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



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



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

Optimal value is the same
But the solution is not unique

Theoretical guarantee for Hungarian algorithm

- Theorem (3.2.11, W) The Hungarian Algorithm finds a maximum weight matching and a minimum cost cover

Back to (unweighted) bipartite graph

- The weights are binary $0,1$
- Hungarian algorithm always maintain integer labels in the weighted cover, thus the solution will always be $0,1$
- The vertices receiving label 1 must cover the weight on the edges, thus cover all edges
- So the solution is a minimum vertex cover

Stable Matchings

Stable matching

- A family $(\leq_v)_{v\in V}$ of linear orderings $\leq_v$ on $E(v)$ is a set of preferences for G
- A matching M in G is stable if for any edge $e \in E \setminus M$, there exists an edge $f \in M$ such that e and f have a common vertex v with $e <_v f$
  - Unstable: There exists $xy \in E \setminus M$ but $xy', x'y \in M$ with $xy' <_x xy$ $x'y <_y xy$

3.2.16. Example. Given men x, y, z, w, women a, b, c, d, and preferences listed below, the matching $\{xa, yb, zd, wc\}$ is a stable matching. ■

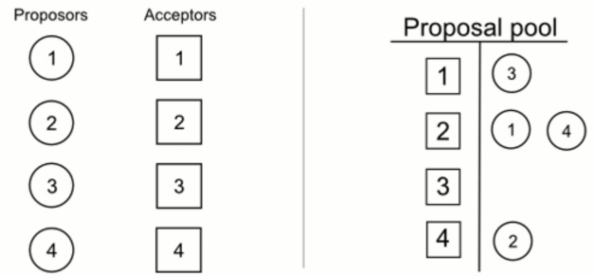
Men $\{x, y, z, w\}$	Women $\{a, b, c, d\}$
$x : a > b > c > d$	$a : z > x > y > w$
$y : a > c > b > d$	$b : y > w > x > z$
$z : c > d > a > b$	$c : w > x > y > z$
$w : c > b > a > d$	$d : x > y > z > w$

Gale-Shapley Proposal Algorithm

- **Input**: Preference rankings by each of n men and n women
- **Idea**: Produce a stable matching using proposals by maintaining information about who has proposed to whom and who has rejected whom
- **Iteration**: Each man proposes to the highest woman on his preference list who has not previously rejected him
  - If each woman receives exactly one proposal, stop and use the resulting matching
  - Otherwise, every woman receiving more than one proposal rejects all of them except the one that is highest on her preference list
  - Every woman receiving a proposal says "maybe" to the most attractive proposal received

Example

Round : 1



- 1-4 propose, as none are currently tentatively attached

Preferences

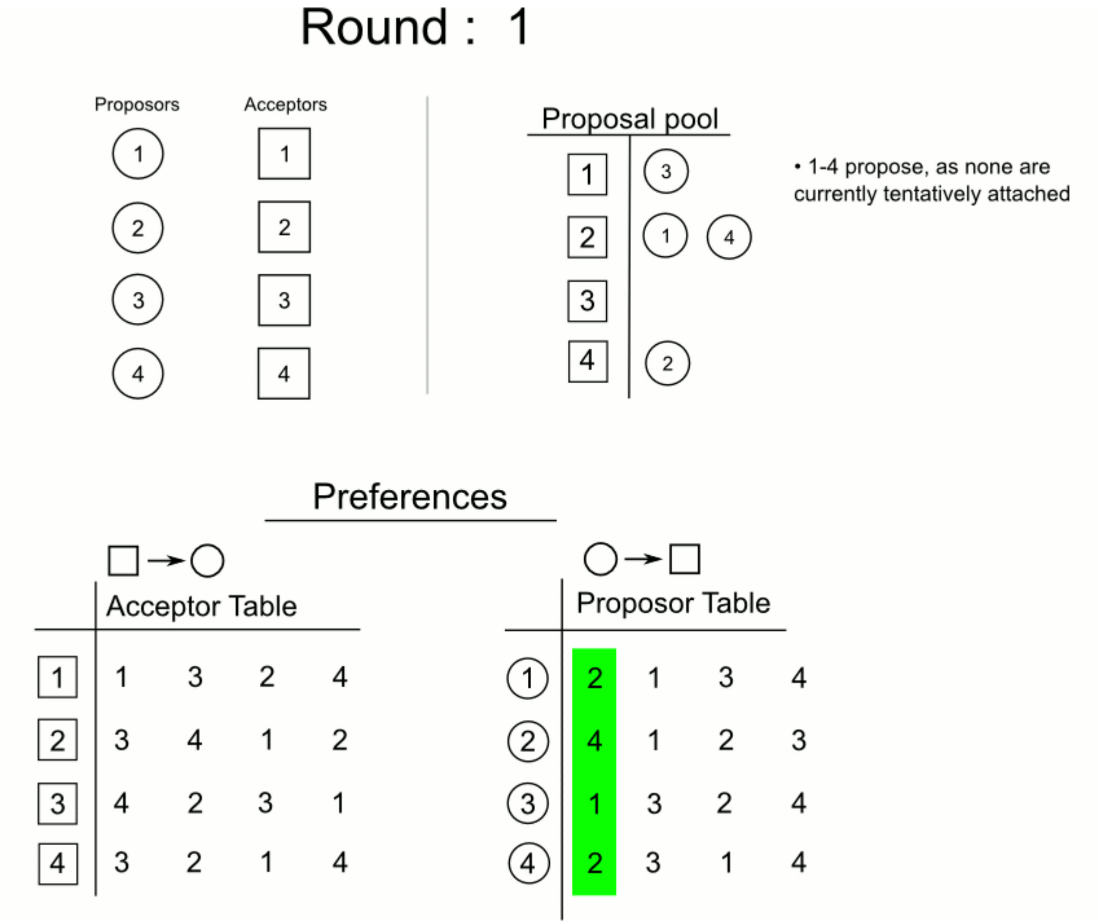
Acceptor Table (□→○)

1	1	3	2	4
2	3	4	1	2
3	4	2	3	1
4	3	2	1	4

Proposor Table (○→□)

1	2	1	3	4
2	4	1	2	3
3	1	3	2	4
4	2	3	1	4

Example (gif)


Round : 1
Proposors
Acceptors
1 2 3 4
1 2 3 4
Proposal pool
1 3
2 1 4
3
4 2
• 1-4 propose, as none are currently tentatively attached
Preferences
Acceptor Table
1 1 3 2 4
2 3 4 1 2
3 4 2 3 1
4 3 2 1 4
Proposor Table
1 2 1 3 4
2 4 1 2 3
3 1 3 2 4
4 2 3 1 4

Theoretical guarantee for the Proposal Algorithm

- Theorem (3.2.18, W, Gale-Shapley 1962) The Proposal Algorithm produces a stable matching
- Who proposes matters (jobs/candidates)
- When the algorithm runs with women proposing, every woman is as least as happy as when men do the proposing
    - And every man is at least as unhappy

3.2.16. Example. Given men x, y, z, w, women a, b, c, d, and preferences listed below, the matching $\{xa, yb, zd, wc\}$ is a stable matching. ■

Men $\{x, y, z, w\}$	Women $\{a, b, c, d\}$
$x : a > b > c > d$	$a : z > x > y > w$
$y : a > c > b > d$	$b : y > w > x > z$
$z : c > d > a > b$	$c : w > x > y > z$
$w : c > b > a > d$	$d : x > y > z > w$

Matchings in general graphs

Perfect matchings

- K_{2n}, C_{2n}, P_{2n} have perfect matchings
- Corollary (3.1.13, W; 2.1.3, D) Every k-regular ($k > 0$) bipartite graph has a perfect matching

- Theorem(1.58, H) If G is a graph of order $2n$ such that $\delta(G) \geq n$, then G has a perfect matching

Theorem (1.22, H, Dirac) Let G be a graph of order $n \geq 3$. If $\delta(G) \geq n/2$, then G is Hamiltonian

Tutte's Theorem (TONCAS)

- Let $q(G)$ be the number of connected components with odd order
- Theorem (1.59, H; 2.2.1, D; 3.3.3, W)
  Let G be a graph of order $n \geq 2$. G has a perfect matching $\Leftrightarrow q(G - S) \leq |S|$ for all $S \subseteq V$

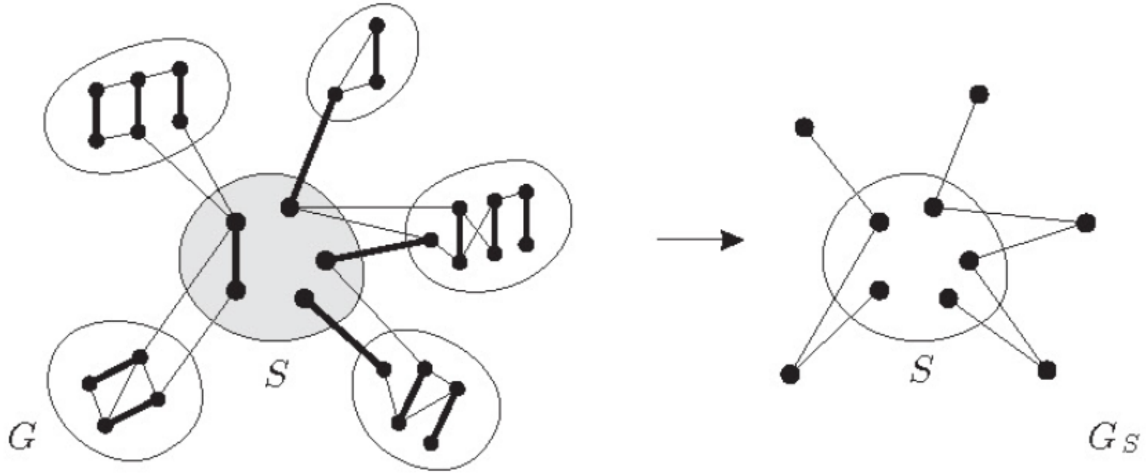


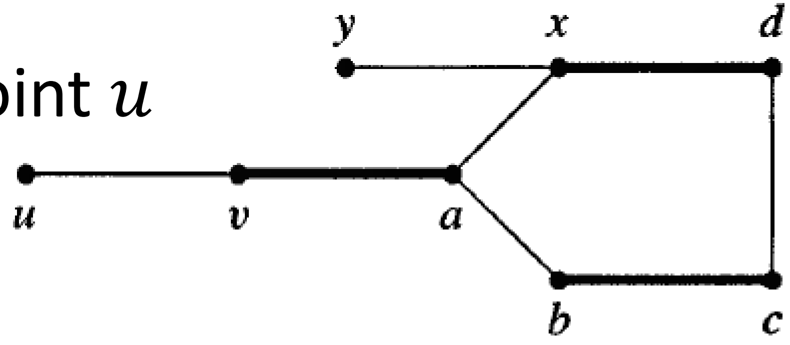
Fig. 2.2.1. Tutte's condition $q(G-S) \leqslant |S|$ for $q = 3$, and the contracted graph G_S from Theorem 2.2.3.

Petersen’s Theorem

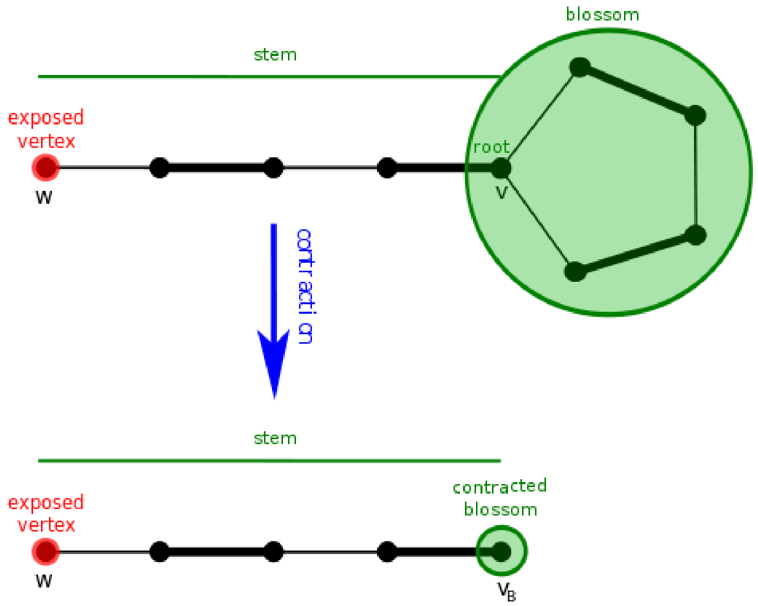
- Theorem (1.60, H; 2.2.2, D;3.3.8, W)
  Every bridgeless, 3-regular graph contains a perfect matching

Find augmenting paths in general graphs

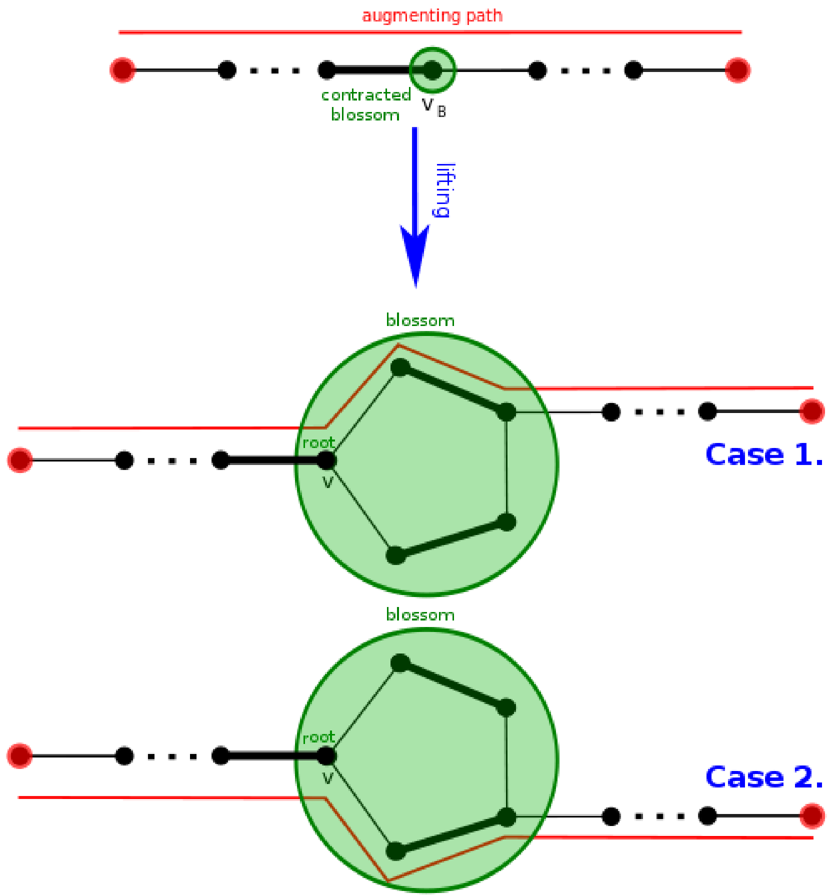
- Different from bipartite graphs
- Example: How to explore from M-unsaturated point u



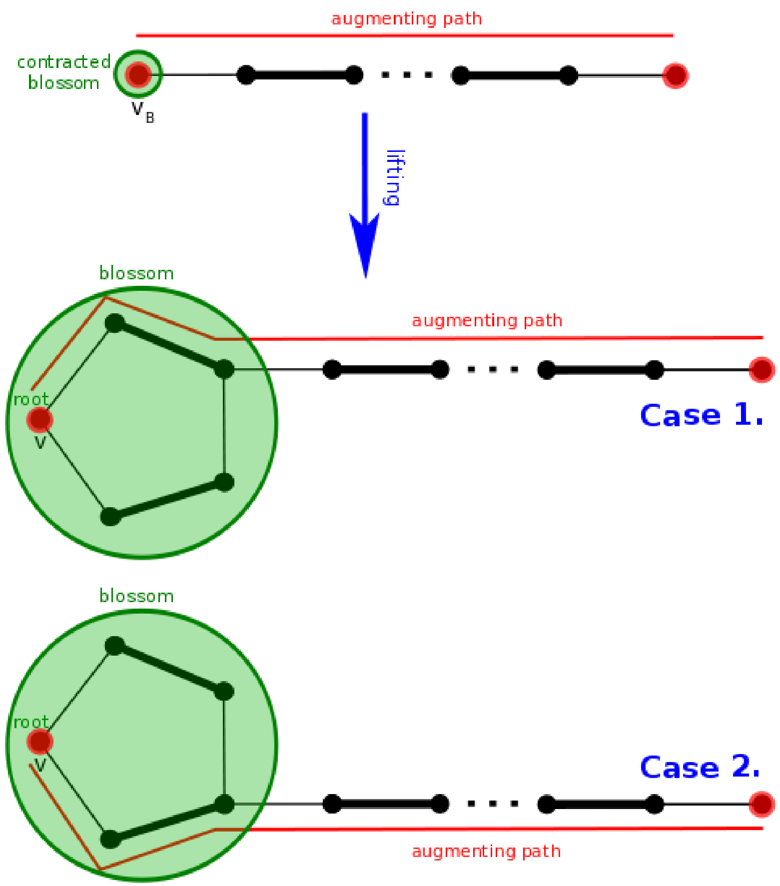
- Flower/stem/blossom



Lifting



augmenting path
contracted blossom
v_B
lifting
blossom
root
v
Case 1.
blossom
root
v
Case 2.

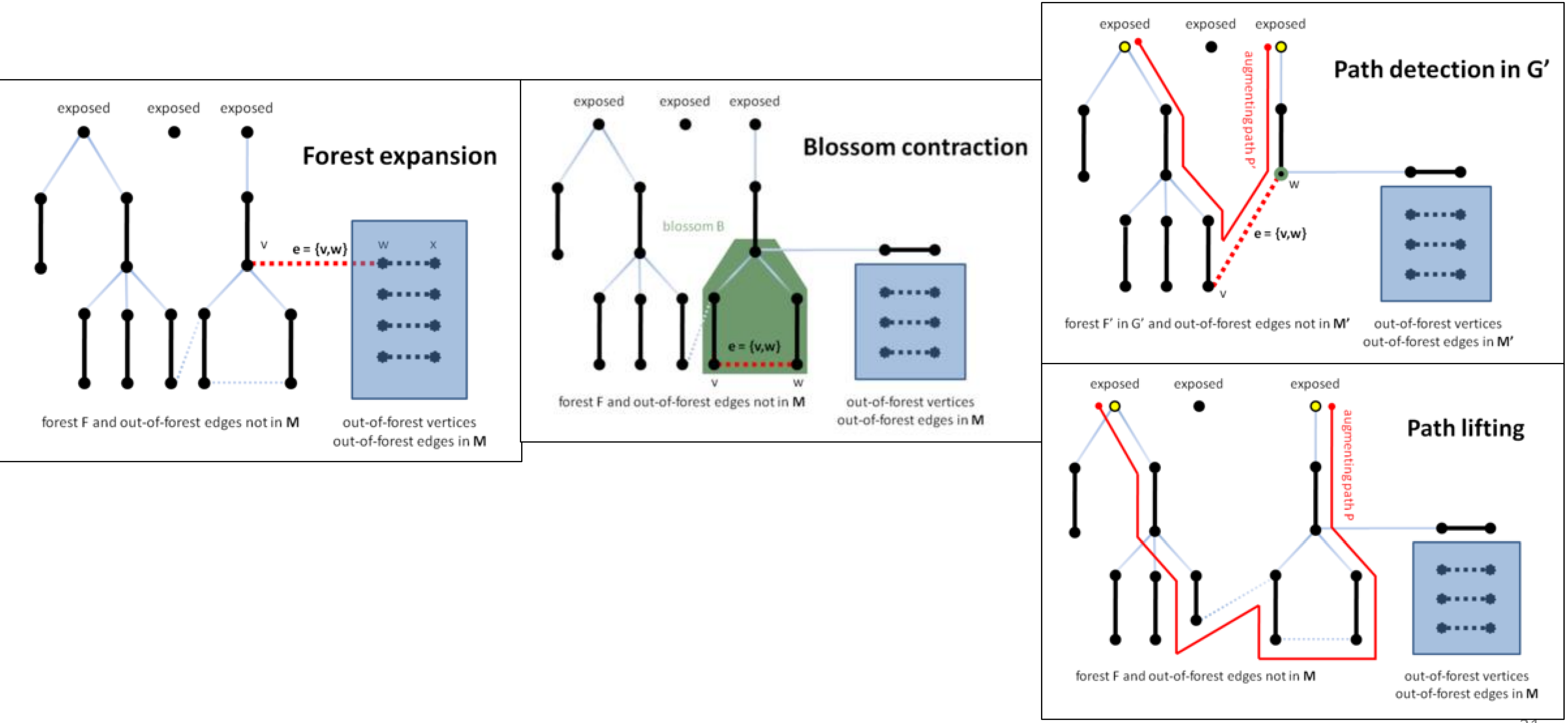


augmenting path
contracted blossom
v_B
lifting
blossom
augmenting path
root
v
Case 1.
blossom
root
v
Case 2.
augmenting path

Edmonds' blossom algorithm (3.3.17, W)

- **Input**: A graph G, a matching M in G, an M-unsaturated vertex u
- **Idea**: Explore M-alternating 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 Augmenting Path Algorithm, with S consisting of u and the vertices reached along saturated edges
  - Reaching an unsaturated vertex yields an augmentation.
- **Initialization**: $S = \{u\}$ and $T = \emptyset$
- **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 $y \in N(v)$ s.t. $y \notin T$
    - If y is unsaturated by M, then trace back from y (expanding blossoms as needed) to report an M-augmenting 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

Illustration



Example