Lecture 3: Weighted Bipartite Matching

Outline

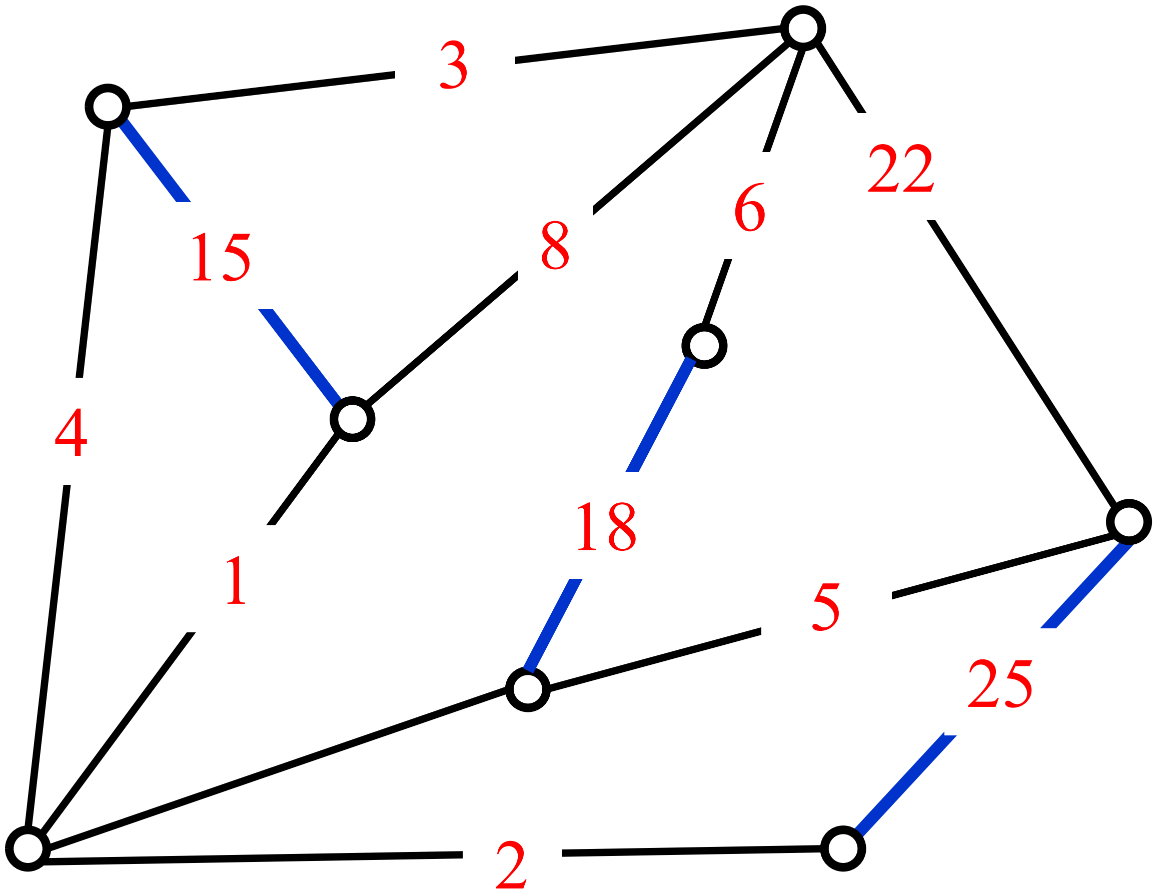
- Weighted matching
- Max-weight bipartite matching
- Stable set, vertex cover, edge cover
- Shortest paths in digraphs

1. Weighted Matching

Weighted matching

- $G = (V, E; \ell)$: an edge-weighted undirected graph
- Max-Weight Matching: find a matching M with maximum weight

$$\ell(M) \coloneqq \sum_{e \in M} \ell(e)$$



Edge lengths w.r.t. a matching

Def. Edge length function ℓ_M w.r.t. a matching M:

$$\ell_M(e) = \begin{cases} \ell(e), & \text{if } e \in M; \\ -\ell(e), & \text{otherwise.} \end{cases}$$

Fact: For any M-alternating path or circuit P, $\ell(M \oplus P) = \ell(M) - \ell_M(P)$

Extreme matching

Def. A matching M is said to be extreme if it has maximum weight among all matchings of size $|M|$.

Claim. If M is extreme, then G has no negative (w.r.t. ℓ_M) M-alternating circuit.

Pf. For any negative alternating circuit C in G, $M \oplus C$ is matching of size $|M|$ and weight larger than M.

Symmetric difference of two extreme matchings

M, N: extreme matchings with $|N| - |M| = k > 0$.

Thm. There exist k vertex-disjoint M-augmenting paths whose total length w.r.t. ℓ_M is $\ell(M) - \ell(N)$.

Claim. Each even component of $M \oplus N$ must have zero length (w.r.t. ℓ_M).
Claim. Each component pair of M-augmenting path and N-augmenting path in $M \oplus N$ must have 0 length (w.r.t. ℓ_M) in total.

Pf. Otherwise, it would be possible to exchange the M and N edges on this component to increase the weight of either M or N.

Growth of extreme matchings

Thm. If M is an extreme matching and P is a shortest M-augmenting path w.r.t. ℓ_M, then $M \oplus P$ is also an extreme matching of size $|M| + 1$.

Pf. For any extreme matching N of size $|M| + 1$, there is an M-augmenting path Q s.t.

$$\ell(N) = \ell(M) - \ell_M(Q) \leq \ell(M) - \ell_M(P) = \ell(M \oplus P) \, .$$

Optimality test

Thm. Suppose that M is a matching of max weight among all matchings of size **at most** $|M|$ and M has no negative M-augmenting path w.r.t. ℓ_M. Then M is a max-weighted matching.

Early termination

Shortest augmenting-path method

$M \leftarrow \emptyset$;
repeat
    find a shortest M-augmenting path P (if any) w.r.t. ℓ_M;
    if P is not found or P has non-negative length, return M;
    $M \leftarrow M \oplus P$;

Fact: all intermediate matchings are extreme, and their weights are strictly increasing

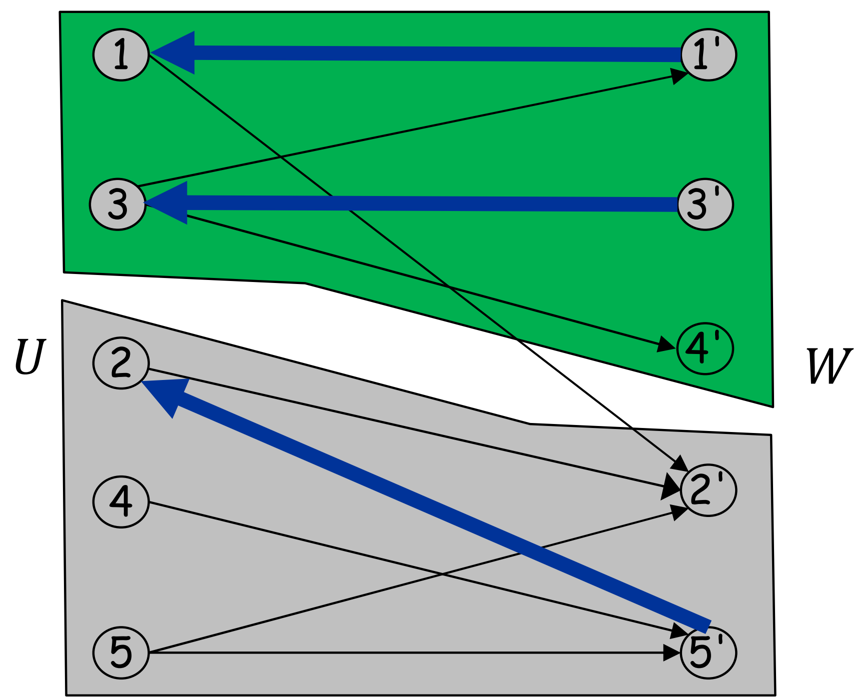
Remark: May be modified to compute extreme matching of given size

Challenge: How to find a shortest augmenting path? Easy in bipartite graph

2. Max-Weight Bipartite Matching

Recap: augmenting graph of a matching

- D_M: edges in M are oriented from W to U, others from U to W
- Each circuit C in D_M is M-alternating
- If M is extreme, then D_M has no negative circuit w.r.t. ℓ_M.



Ford-Fulkerson Algorithm

Shortest M-augmenting path: compute a shortest $U_M - W_M$ path in D_M using Bellman-Ford algorithm

Analysis. $O(n^2m)$ time: $O(n)$ augmentations, each taking $O(nm)$ time

Hungarian Algorithm

Shortest M-augmenting path: compute a shortest shortest $U_M - W_M$ path in D_M using Dijkstra's algorithm with potential

Analysis. O(n(m+nlog n)) time: O(n) augmentations, each taking O(m+nlog n) time

Distance-based potentials

M: an extreme matching;
P: a shortest $U_M - W_M$ path in D_M;
$p_M(v) :=$ distance from U_M to v in D_M, for each $v \in R_M$.

Claim. For $N := M \oplus P$, p_M is a potential for $D_N[R_N]$

Pf. Consider an arc (u, v) of $D_N[R_N]$.
Case 1. (u, v) is also an arc of D_M. Then

$$p_M(v) \le p_M(u) + \ell_M(u, v) = p_M(u) + \ell_N(u, v)$$

Case 2. (u, v) is not an arc of D_M. Then its reverse $(v, u) \in P$, and hence

$$p_M(v) = p_M(u) - \ell_M(u, v) = p_M(u) + \ell_N(u, v)$$

Initial extreme matching and potentials

- e ≔ an edge of maximum weight
- initial extreme matching $M \coloneqq \{e\}$
- a potential p in D_M: (verification as exercise)

$$p(v) = \begin{cases} 0, & if\ v \in U; \\ -\max_{(u,v)\in E} w(u,v), & if\ v \in W. \end{cases}$$

Implementation

```
Hungarian-WBM(G)

   M ← {an edge e of maximum weight}
   foreach v ∈ U, p(v) ← 0
   foreach v ∈ W, p(v) ← −max_{(u,v)∈E} w(u,v)

   repeat
        reweight all arcs in D_M using p
        apply Dijkstra's algorithm to compute P and p
        if P is not found or has non-negative length,
           return M
        else M ← M ⊕ P
```

3. Stable set, Vertex Cover, Edge Cover

Stable set, vertex cover

$G = (V, E;\ w)$: a vertex-weighted undirected graph

Maximum weight stable set
Minimum weight vertex cover

NP-hard in general graphs
Polynomial in bipartite graphs: reduction to Min-Cut

Edge cover

$G = (V, E;\ w)$: an edge-weighted undirected graph without isolated vertices

Minimum-weight edge cover: Polynomial in general graphs

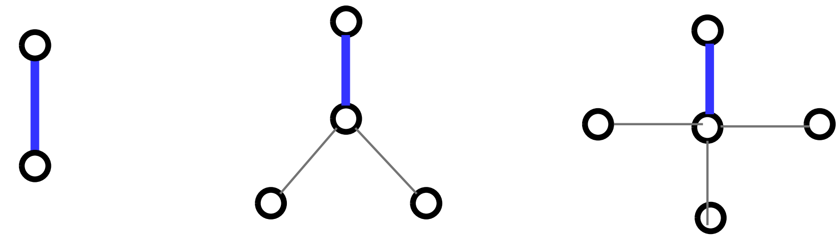
Observation: All edge weights can be assumed to be positive

Include all edges with non-positive weights, and then remove them and their ends. Proceed to the remaining graph

Structure of a min-weight edge cover

Let F be a min-weight edge cover.

- (V, F) is a forest of stars.



- In each star,
  - at most one leaf is incident to the edge whose weight is not the least among all its incident edges.
  - if such leaf exists, its incident edge in the star is chosen as a matched edge; otherwise, choose an arbitrary edge in the star as the matched edge

- All matched edges form a matching M; all other edges in F are min-weight edges incident to vertices missed by M

Weight of min-weight edge cover

- For each $v \in V$, $c(v) := $ min. weight of edges incident to v
- For each $(u, v) \in E$, $\ell'(u, v) := c(u) + c(v) - \ell(u, v)$

Claim. $\ell(F) = c(V) - \ell'(M)$

pf.

$$\begin{aligned} \ell(F) &= \ell(M) + c\big(V \setminus V(M)\big) \\ &= c(V) - c\big(V(M)\big) + \ell(M) \\ &= c(V) - \ell'(M) \end{aligned}$$

Algorithm for min-weight edge cover

$F \leftarrow M \leftarrow$ a maximum-weight matching of G w.r.t. ℓ';
for each v missed by M
    add to F an min-weight edge incident to v;
return F.

Correctness. $\ell(F) = c(V) - \ell'(M)$

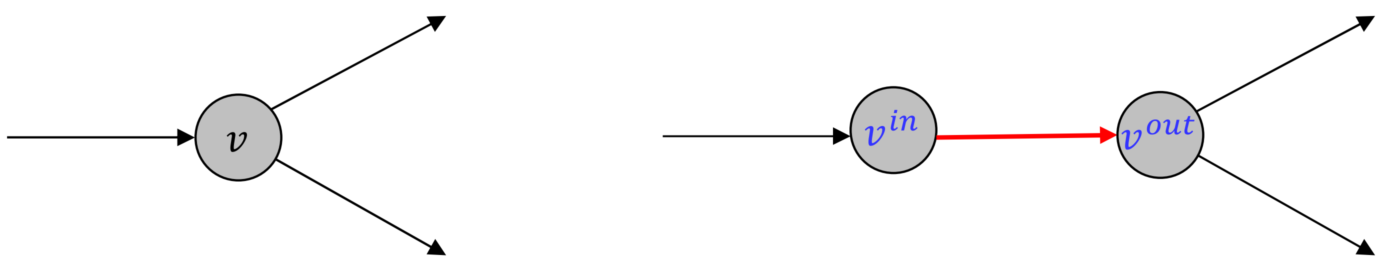
3. Shortest Path in Digraphs

Node splitting

$D = (V, A; \ell)$: digraph with arbitrary edge length

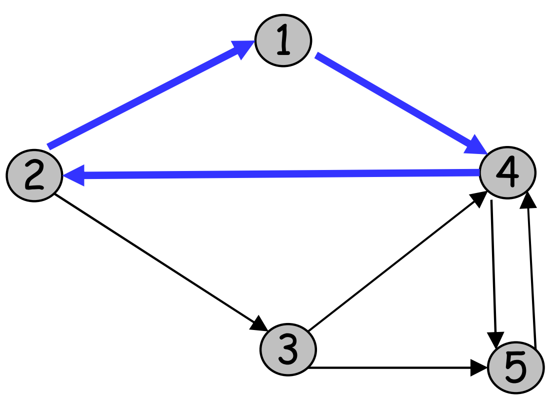
G via splitting of each node v:

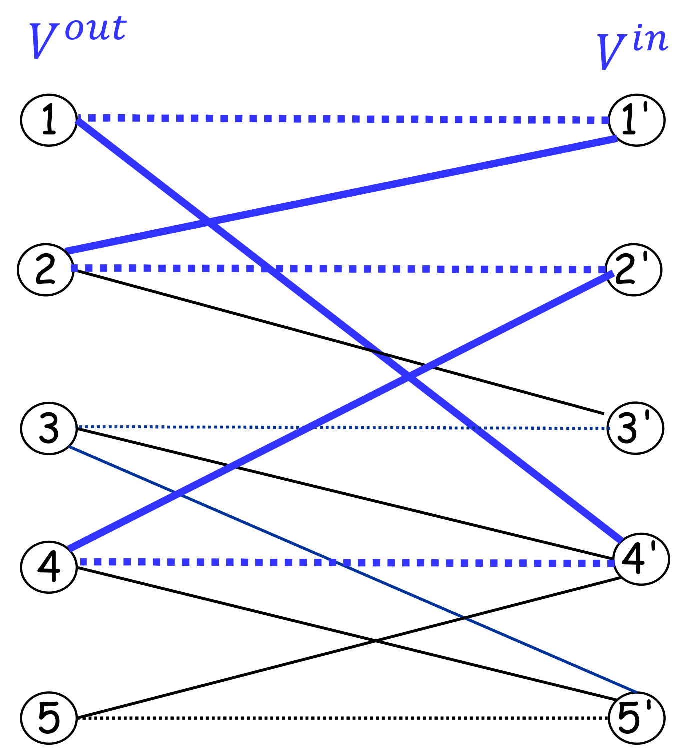
- replace v by a 0-length self-edge (v^{in}, v^{out});
- each edge entering v now enters v^{in};
- each edge leaving v now leaves v^{out};
- ignore the directions



Bipartite graph

- bipartite between V^{in} and V^{out}
- all self-edges form a perfect matching M_0 of 0 weight
- circuit C in D $\leftrightarrow$ M_0-alternating circuit C^+ in G





Detection of negative circuit

Fact. $M_0 \oplus C^+$ is a perfect matching with weight $\ell(C)$.

Fact. For any perfect matching M, all components of $M_0 \oplus M$ are circuits whose total length is the weight of M.

Thm. Suppose M is a min-weight perfect matching.

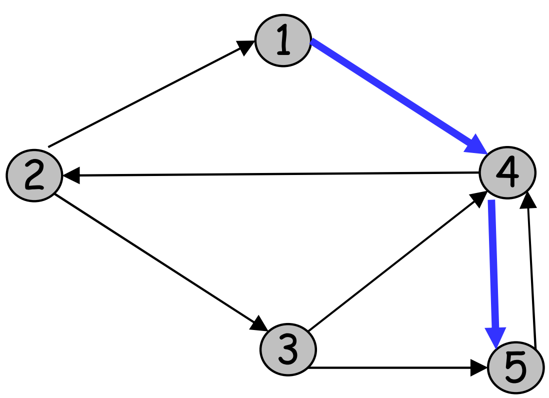
- If M has 0 weight, then all circuits in D are non-negative;
- otherwise, a negative component of $M_0 \oplus M$ corresponds to a negative circuit in D.

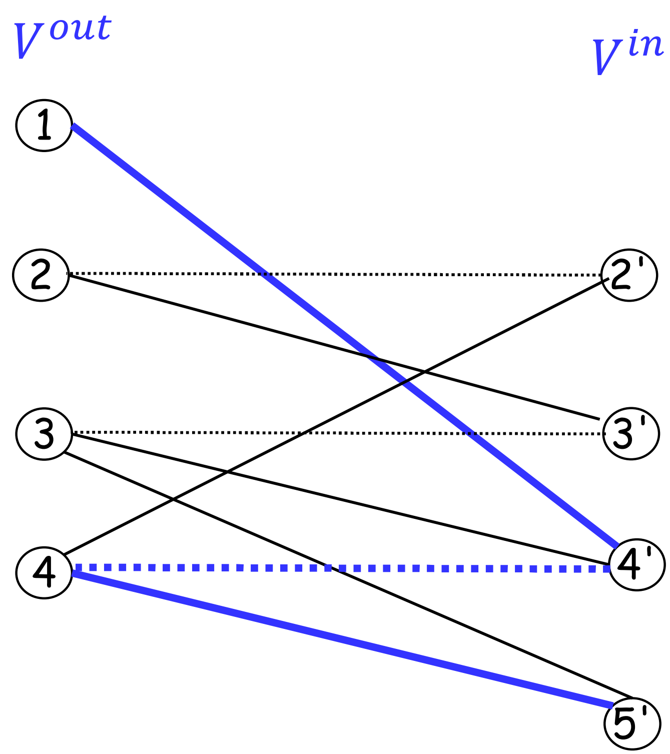
Shortest s-t path in case of no negative circuit

Remove s^{in} and t^{out} from G and M_0: $|V^+| = 2(n-1), |M_0| = n-2$

circuit C in $D - \{s,t\}$ $\leftrightarrow$ M_0-alternating circuit C^+ in G

s-t path P in D $\leftrightarrow$ M_0-augmenting s^{out} - t^{in} path P^+ in G





Shortest s-t path in case of no negative circuit

Fact. $M_0 \oplus P^+$ is a perfect matching (of size $n-1$) with weight $\ell(P)$.

Fact. For any perfect matching M, among all components of $M_0 \oplus M$, one is an M_0-augmenting s^{out} - t^{in} path and all others are M_0-alternating circuits.

Thm. Suppose M is a min-weight perfect matching. Then the path component of $M_0 \oplus M$ corresponds to a shortest s-t path in D.

Summary

- Augmenting paths, extreme matchings
- Hungarian algorithm: one stone, two birds
- Stable set, vertex cover, edge cover
- Shortest paths in digraphs