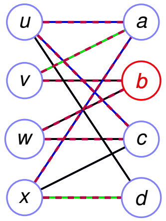
Matching.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.

A matching is a set of edges where no two share an endpoint.



Blue – 3. Green - 2, Black - 1, Non-edges - 0.

Solution Value: 7.

Solution Value: 7.

Solution Value: 8.

Applications

Jobs to workers.

Teachers to classes.

Classes to classrooms.

"The assignment problem"

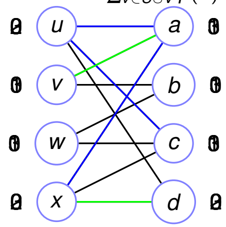
Min Weight Matching.
  Negate values and find maximum weight matching.

Vertex Cover

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find an vertex cover function of minimum total value.

A function $p : V \to R$, where for all edges, $e = (u, v)$ $p(u) + p(v) \geq w(e)$.

Minimize $\sum_{v \in U \cup V} p(u)$.



Solution Value: 12.

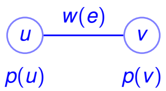
Solution Value: 12.

Solution Value: 9.

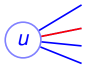
Solution Value: 8.

Cover is upper bound.

Feasible $p(\cdot)$, for edge $e = (u, v)$, $p(u) + p(v) \geq w(e)$.



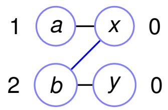
For a matching M, each u is the endpoint of at most one edge in M.



$$\sum_{e=(u,v) \in M} w(e) \leq \sum_{e=(u,v)} (p(u) + p(v)) \leq \sum_{u \in U} p(u) + \sum_{v \in V} p(v)$$

Holds with equality if
  for $e \in M$, $w(e) = p(u) + p(v)$ (Defn: tight edge.) and
  perfect matching.

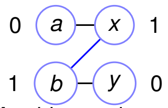
Simple example.



Blue edge – 2, Others – 1.

Using max incident edge.

Value: 3.
Using max incident edge.



Value: 2.
  Same as optimal matching!

Proof of optimality.

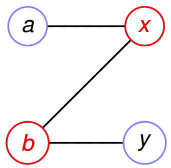
Matching and cover are optimal,
  edges in matching have $w(e) = p(u) + p(v)$. Tight edge.
  all nodes are matched.

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

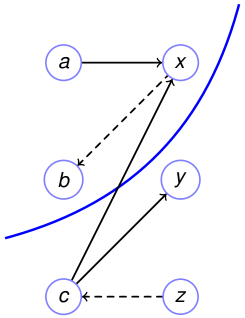
Key Idea: Augmenting Alternating Paths.

Example:



Start at unmatched node(s),
  follow unmatched edge(s),
    follow matched.
      Repeat until an unmatched node.

No perfect matching



Can't increase matching size.
  No alternating path from (a) to (y).

Cut!

Still no augmenting path.
  Still Cut?

Use directed graph!
  Cut in this graph.

Algorithm:
Given matching.
Direct unmatched edges U to V, matched V to U.
Find path between unmatched nodes on left to right. (BFS, DFS).
Until everything matched ... or output a cut.

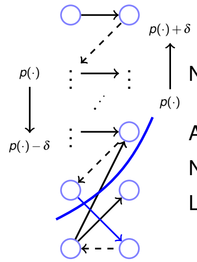
Back to Maximum Weight Matching.

Want vector cover (price function) $p(\cdot)$ and matching where.

Optimal solutions to both if
  for $e \in M$, $w(e) = p(u) + p(v)$ (Defn: tight edge.) and
  perfect matching.

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Start with empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
  Use alt./aug. paths of tight edges.
    "maximum matching algorithm."

No augmenting path.
  Cut, (S,T), in directed graph of tight edges!

All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from S_U, T_V.

Lower prices in S_U, raise prices in S_V,
  all explored edges still tight,
  backward edges still feasible

... and get new tight edge!
What's delta? $w(e) > p(u) + p(v) \rightarrow$
$\delta = min_{e \in (S_U \times T_V)} w(e) - p(u) - p(v)$.

Lecture 2 ended in the middle of the previous slide.
A question was asked why don't we just drop prices around the blue (loose) edge in the figure. Why not?

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

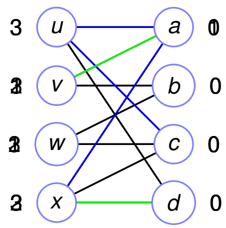
Beginning "Coverer" Solution: $p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

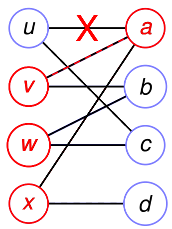
Main Work:
  breadth first search from unmatched nodes finds cut.
Update prices (find minimum delta.)

Simple Implementation:
  Each bfs either augments or adds node to S in next cut.
  $O(n)$ iterations per augmentation.
  $O(n)$ augmentations.

$O(n^2 m)$ time.

Example





All matched edges tight.
Perfect matching. Feasible price function. Values the same.
Optimal!

Notice:
  no weights on the right problem.
  retain previous matching through price changes.
  retains edges in failed search through price changes.