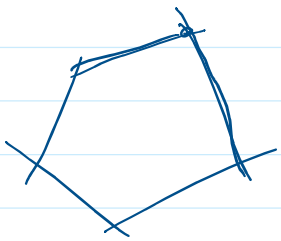
Further notes about LP's

Suppose P is a polyhedron in $\mathbb{R}^n$ defined by set of inequalities $Ax \leq b$. If P is finite and feasible then every bfs (basic feasible solution) corresponds to a set of n linearly independent constraints that all hold with equality, i.e. are "tight."

In other words, if we have an LP with n var's

$$\max c^T \cdot x$$
$$A \cdot x \leq b$$
$$x_i \geq 0$$



then if it is bounded & feasible then for every bfs x^* there is a set of n linearly independent constraints $a_{ij} \cdot x_i \leq b_j$ $\quad j \in I \subseteq \{1 \ldots m\}$ where x^* is the unique solution to the set of equations $a_{ij} \cdot x_i = b_j$, $j \in I$

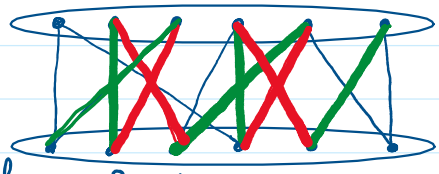
Combinatorial Algorithms for Matching

- We have already seen that for any v.c. C & any matching M: $|C| \geq |M|$

Theorem (König): For bipartite G: $\min_{\text{v.c. } C} |C| = \max_{\text{matchg } M} |M|$

- We have already seen a proof via LP.

Definition: Given matching M, an M-alternating path is a path that alternates between M & $E-M$



M

M-augmenting: if the first & last vertex are not saturated (are exposed).

Lemma: if M is a matching & P is an M-augmenty then $M \Delta P = (M-P) \cup (P-M)$ is a matching of size $|M|+1$.

Proof: Easy.

Theorem: A matching M is maximum iff there is no M-augmenting path.

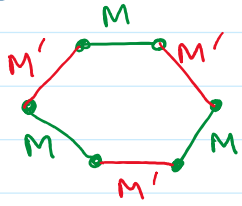
Proof: One direction is easy.

- suppose there is no M-augmenting path but there is a matching M' with $|M'| > |M|$.
- Consider graph $H = (A \cup B, M \cup M')$.

Fact: H has max degree 2.

$\longrightarrow$ each component is { a path or cycle.

- Each cycle C is even:
- $\exists$ a path (alternating)



P with $|P \cap M'| > |P \cap M|$

$\longrightarrow$ P is an M-augmenting path.

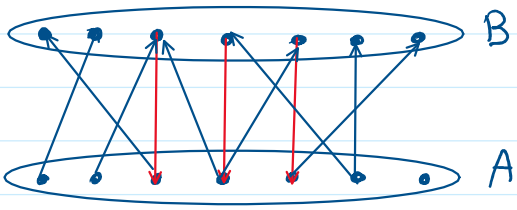
Bipartite Matching & Vertex Cover

- Based on M-augmenting paths we can build an algorithm to find maximum matching

> Maximum bipartite Matching
> $M \leftarrow \phi$
> while the is an M-augmenting path P do
> let $M \leftarrow M \Delta P$
> return M

- We provide an $O(m)$ algorithm that finds an M-augmenting path.
- Imagine we make the graph directed $D=(A \cup B, E')$
  - Direct $e \in M$ from B to A
  - Direct $e \notin M$ from A to B

Lemma: There is an M-augmenting path in G iff $\exists$ a dipath from an exposed node in A to an exposed node in B.



Proof: Easy excercise.

Consider an extra node r with di-edges to exposed nodes in A. then run a BFS from r to find a path to exposed nodes in B. ∎

Theorem: We can find a maximum matching in time $O(mn)$.

Proof: Size of matching is $O(n)$; finding an M-augmenting path takes $O(m)$.

improved time: If one uses BFS to find "shortest" M-augmenting paths then it can be shown that one can find vertex disjoint M-augmenting paths

- So can increase $|M|$ by more than 1.

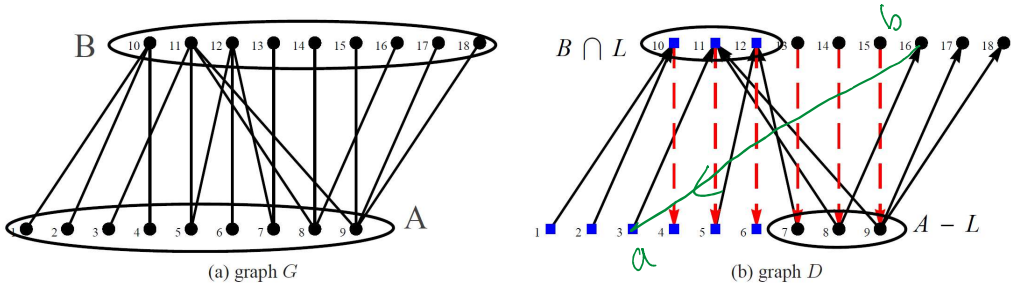
Theorem: Using BFS one can find maximum matchig in time $O((m+n)\sqrt{n})$.

Fastest: A breakthrough result in FOCS 22 (112 pages!) show bipartite matching in time $O(m^{1+o(1)})$. (Chen et al. FOCS 22).

What about min V.C. for bipartite G?

Consider the digraph $D=(A \cup B, E')$ where edges of M are directed from B to A (and $e \in E-M$ from A to B).

- let L be the set of vertices that can be reached from an exposed node of A



(a) graph G (b) graph D

Lemma: When the matching algorithm terminates $C^* = (A-L) \cup (B \cap L)$ is a vertex cover and $|C^*| = |M|$.

proof: Suppose C^* is not a V.C. $\longrightarrow$

$\exists$ an edge $ab \in E$ s.t. $a \in A \cap L$ & $b \in B-L$

claim: $ab \notin M$

proof: Suppose not, so $ab \in M$ and is directed $b \to a$; since $a \in L$, it can be reached only by ba (only edges in M) from an exposed node, (note that a itself is not exposed as $ab \in M$) so b is also reached from the exposed node

$\longrightarrow b \in L$ but $b \in B - L$ ✗

So $ab \in E - M \longrightarrow$ directed from a to b

Since $a \in L$ and ab is an edge so b is reachable from an exposed node $\longrightarrow b \in L$ ✗

So $e \notin M \wedge e \notin E - M$, So C^* is V.C.

Prove $|C^*| \leq |M|$:

- Note no vertex in $A - L$ is exposed (since exposed vertices of A are in L).
- No $b \in B \cap L$ is exposed (or else $\exists$ an M-augmenting path).
- Also we showed $\nexists ab \in E$ with $a \in A - L$, $b \in B \cap L$

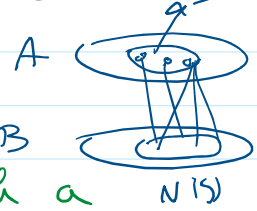
$\longrightarrow |M| \geq |(A - L) \cup (B \cap L)| = |C^*|$ ∎

- This gives another proof of König's theorem.
- We can prove the following using König's theorem

Theorem (Hall 1935): A bipartite graph $G = (A \cup B, E)$ has a matching that saturates all of A iff

$\forall S \subseteq A: |N(S)| \geq |S|$

$N(S)$ = vertices in B with a



neighbor in S

Weighted Bipartite Graph Matching

Problem: $G(A \cup B, E)$, $w: E \to \mathbb{R}^+$, find max-weight matching.

Assumptions:

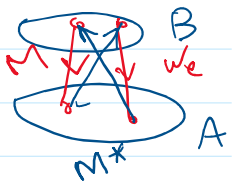
- G is complete bipartite (add 0-weight edges)
- $|A| = |B|$ (add dummy vertices & 0-edges)
- $w_e \geq 0$ (if not add $W = \max_e |w_e|$ to all)

minimization: Can define $c_e = W - w_e$

- we use $w(M)$: weight of matching M.

1) First Algorithm: Negative cycles.

- Build digraph $D = (A \cup B, E')$ as before: $\begin{cases} e \in M & B \text{ to } A \\ e \notin M & A \text{ to } B \ -w_e \end{cases}$
- let M be the current matching (M is a perfect matching always).



- if $w(M^*) > w(M)$ then $H = (A \cup B, M \cup M^*)$ is union of cycles $\longrightarrow \exists$ a negative cycle!

$$w(H) = \sum_{C \in \mathcal{C}} w(C) = w(M) - w(M^*)$$

$\mathcal{C}$ → set of cycles in H

– We can use Bellman-Ford Shortest path to find one in $O(mn)$ time; swap edges.

Maximum Weight bipart Matching: Algorithm 1

$M \leftarrow$ any perfect matching

Build D from M

while there is a negative cycle C do

$M \leftarrow M \Delta C$

update M

return M

Theorem: A perfect weighted matching M is optimum iff there is no negative cycle C.

Proof: One direction is easy.

Suppose M is not maximum & $\nexists$ negative cycle C

– let M' be a max-weight matching & build $D' = M \cup M'$

– D' is union of disjoint edges & cycles

$$C \subseteq (M \cup M') - (M \cap M')$$

– Note: $w(M' - M) > w(M - M')$

$\rightarrow$ there is a negative cycle!

Time Complexity: $O(mn \times (w_{max} - w_{min}))$; inefficient

Goldberg & Tarjan '89: if one finds negative cycles with

min average weight $\frac{w(C)}{|C|}$ ⟶ time complexity will be strongly-polynomial.

2) Second Algorithm: Hungarian Method

Developed by Kuhn (1955) using Egerváry's theorem.

later improved by Munkres'57, Iri'60, Edmond & Karp'70

- At each step $\underline{k}$ it has a max-weight matching with exactly $\underline{k}$ edges.
- Start with $M = \phi$
- In each iteration, build D (as before):

$e \in M$	directed B to A	
$e \notin M$	" A to B	with $-w_e$

- If there is an M-augmenting path P then

$M' = M \Delta P$: $w(M') = w(M) - w(P \cap M) + w(P - M)$

$= w(M) - \ell(P)$

$\ell(P) = w(P \cap M) - w(P - M)$

$|M'| = |M| + 1$; so $k+1$

Theorem: Augmenting a max-matching of size k by a shortest augmenting path produces a max matching of size $k+1$.

Proof: Induction on k: $k \geq 0$

- Suppose M is a max-matching of size k & P a shortest M-augmenting path
- Suppose $M' = M \Delta P$ is not maximum; let N be a maximum size $k+1$ matching: $w(N) > w(M')$
- let $H = (A \cup B, M \cup N) \subseteq D$; Since $|N| > |M|$, $\exists$ an M-augmenting path P' in H.
- Since P is the shortest M-augmenting in D and $H \subseteq D$:

$$\ell(P') \geq \ell(P) \qquad (1)$$

- Consider $N' = N \Delta P'$ obtained by applying P' in reverse to N ($|N'| = k$).

$$w(N') \leq w(M) \qquad (2)$$

this is because N' is a matching of size k and M is the max. matching of size k.

- From (1) & (2): $\underbrace{w(N') - \ell(P')}_{w(N)} \leq w(M) - \ell(P)$

$$w(N) \leq w(M').$$

Max_weight bipartite Matching: Hungarian Method

$M \leftarrow \arg\max_{w_e} \{e\}$

Build D

while $|M| < \frac{n}{2}$ do

find shortest path P in D

$M \leftarrow M \Delta P$

update D

return M

- Using Bellman_Ford takes $O(mn)$ for shortest path

$\rightarrow$ Total time complexity: $O(mn^2)$

Matching in General Graphs

The Matching (and weighted matching) problem when $G(V,E)$ is not bipartite is more complex.

one can write an LP relaxation which is integral for general graphs.

There are also polynomial time algorithms to find max matching in general graphs. Edmond presented the first such algorithm.

There are also polytime alg. to compute min (or max) weight

matching in general graphs.