Primal/dual Method for weighted Matching

We see a different Primal/dual based algorithm, called Hungarian method, for weighted bipartite matching.

$$\max \sum w_e x_e \qquad \min \sum_{v \in A \cup B} y_v$$

$$\forall\ a \in A \quad \sum_{a \in e} x_e \leq 1 \quad (P) \qquad \forall\ e=ab \quad y_a + y_b \geq w_e \quad (D)$$

$$\forall\ b \in B \quad \sum_{b \in e} x_e \leq 1 \qquad y_v \geq 0$$

$$x_e \geq 0$$

By duality, if $\vec{x}$ and $\vec{y}$ are feasible solutions to (P) and (D) then $w(y) \geq w(M)$ (where y is the vertex cover & M is the matching defined by $\vec{x}$ & $\vec{y}$)

The P/D algorithm starts with a trivial Primal solution i.e. $M = \phi$ (so $x_e = 0$) and a trivial dual solution

$$\forall\ a \in A: \quad y_a = \max_{ab \in E} w(ab)$$

$$\forall\ b \in B: \quad y_b = 0$$

- At each iteration of the algorithm, we build an equality graph G_y as defined below

$$G_y = (A \cup B, E_y): \quad ab \in E_y \iff y_a + y_b = w_e$$

let us call $(y_a + y_b - w_e)$ as the excess of ab.

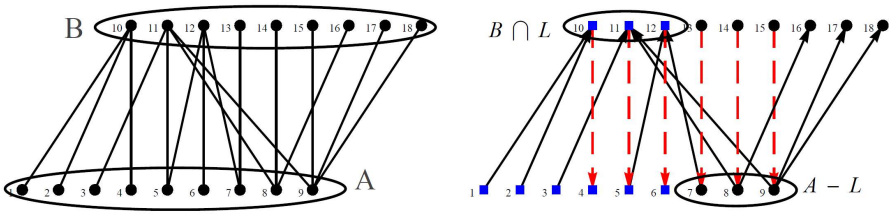
Observation: if M is a perfect matching in G_y then $w(M) = \sum_{a \in A} y_a + \sum_{b \in B} y_b$ and M is an optimum matching

- So the goal of the algorithm will be to find a matching in equality graph G_y.

- So the goal of the algorithm will be to find a matching in equality graph G_y.
- We update y so that more edges go tight to be added to G_y (until it has a perfect matching) while ensuring y remains a vertex cover.
- Suppose we are at some iteration of the algorithm and M is a matching in G_y but not perfect
- We construct the digraph D as before

  $e \in M$: directed B to A

  $e \notin M$: directed A to B



(a) graph G (b) graph D

- let L be the set of nodes accessible from any exposed node in A.
- Recall $C^* = (A - L) \cup (B \cap L)$ is a vertex_cover

  $\longrightarrow$ $\nexists$ no edge between $A \cap L$ and $B - L$
- But G was a complete graph; so there are edges in G but not in G_y

  $\longrightarrow$ those edges have positive excess.
- We update y s.t. one of these edges goes tight

let $\varepsilon = \min \{ y_a + y_b - w_e \text{ s.t. } a \in A \cap L, b \in B - L \}$ be the min excess of all such edges (these are those that can be added to G_y)

those that can be added to G_y)

- update all y's by ε:

$$y_a = \begin{cases} y_a & a \in A - L \\ y_a - \varepsilon & a \in A \cap L \end{cases} \qquad y_b = \begin{cases} y_b & b \in B - L \\ y_b + \varepsilon & b \in B \cap L \end{cases}$$

Note: all edges in G_y remain tight.
No constraint goes violated.
and at least one edge between $A \cap L$ & $B - L$ goes tight $\longrightarrow$ added to G_y.

- repeat this unti G_y has a perfect matching or there is no edges left between $A \cap L$ & $B - L$.

- The latter can happen only if both $A \cap L$ & $B - L = \phi$

$\longrightarrow$ We eventually find a perfect matching in G_y

Max-Weight bip. (perfect) matching: P/D

For each $a \in A$ let $y_a = \max_{b \in B} w(ab)$
For each $b \in B$ let $y_b = 0$
Build G_y and let M be a max matching in G_y
Construct D
repeat
  let L be the nodes reachable from an exposed node of A
  let $\varepsilon = \min \{ y_a + y_b - w(do) \quad a \in A \cap L, \; b \in B - L \}$
  Decrease y_a for all $a \in A \cap L$ by ε
  increase y_b " " $b \in B - L$ " "
  Add tight edges to G_y and recompute M
until M is a perfect matching

Add tight edges to G_y and recompute L

until M is a perfect matching

Runtime: At each iteration one edge is added to G_y $\longrightarrow$ $O(n^2)$ iterations. To compute L & recompute M we spend at most $O(n^2)$ $\longrightarrow$ total time $O(n^4)$

- with more careful analysis the time can be improved to $O(n^3)$.

Primal / Dual for Bipartite Matching

- We have already seen a number of algorithms for bip. matching
- One can design algorithm using LP and dual LP that avoid solving the LP (are combinatorial & fast).
- Hungarian method is one such alg.
- We see another method that finds a $(1-\varepsilon)$-approximate matching in time $O(m/\varepsilon)$; it also finds an approximate vertex cover.
- Suppose we have a bipartite graph $G = (A \cup B, E)$

$$\max \sum_{e \in E} x_e \qquad\qquad \min \sum_{u \in A \cup B} y_u$$

$$\forall\, a \in A: \ \sum_{a \in e} x_e \leq 1 \qquad\qquad \forall\, ab \in E: \ y_a + y_b \geq 1$$

$$\forall\, b \in B: \ \sum_{b \in e} x_e \leq 1 \qquad\qquad y_u \geq 0$$

$$x_e \geq 0$$

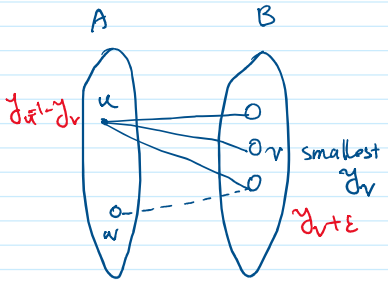
- The algorithm maintains a primal feasible solution (i.e. a matching) and a dual infeasible solution.
- At each iteration, we try to improve optimality of the primal and feasibility of the dual.

$(1-\varepsilon)$-approximate Primal/Dual Matching

- Start from $M=\phi$ and $y_u=0$ (for all u)
- let $U=A$ be the set of unmatched node of A
- while $U\neq\phi$ do
  - * Pick a node $a\in U$ and remove from U
  - * let $b\in B$ be a neighbor of a with smallest y_b
  - * if $y_b=1$ Skip to next iteration
  - * else let a' be the matched neighbor of b if it exists
  - * Add ab to M (a' becomes unmatched and add to U)
  - * update $y_b \leftarrow y_b+\varepsilon$, $y_a=1-y_b$, $y_{a'}=0$
- output M as a matching and $(\frac{y}{1-\varepsilon})$ as a V.C

$(1-\varepsilon)$-approximate Primal/Dual Matching

- Start from $M=\phi$ and $y_u=0$ (for all u)
- let $U=A$ be the set of unmatched node of A
- while $U\neq\phi$ do
  - * Pick a node $a\in U$ and remove from U
  - * let $b\in B$ be a neighbor of a with smallest y_b
  - * if $y_b=1$ Skip to next iteration
  - * else let a' be the matched neighbor of b if it exists
  - * Add ab to M (a' becomes unmatched)
  - * update $y_b \leftarrow y_b+\varepsilon$, $y_a=1-y_b$, $y_{a'}=0$
- output M as a matching and $(\frac{y}{1-\varepsilon})$ as a V.C



Fact 1: if $ab\in M$ then $y_a=1-y_b$, if a is not matched $y_a=0$

Fact 2: $\forall\, b\in B$: $y_b>0 \iff b$ is matched

proof: when b is matched for the first time y_v increases from 0 and it remains matched.

Fact 3: For any $ab\in M$: $y_a+y_b=1$ (so dual const tight)

Lemma 1: At any iteration: $|M|=\sum_{a\in A} y_a+\sum_{b\in B} y_b$

proof: $\sum_{a\in A} y_a+\sum_{b\in B} y_b=\sum_{ab\in M}\underbrace{(y_a+y_b)}_{1}+\underbrace{\sum_{a \text{ not matched}} y_a}_{0 \text{ by Fact 1}}+\underbrace{\sum_{b \text{ not matched}} y_b}_{0 \text{ by Fact 2}}$

$=|M|$ ∎

o by fact 1 ~ by fact 2

Lemma 2: $y/(1-\varepsilon)$ will be a dual feasible solution

Corollary: The algorithm returns a matching M that is $(1-\varepsilon)$-approximate and $y/(1-\varepsilon)$ is a fractional vertex cover.

proof: Note that

$$\begin{aligned} opt_D &\geq opt_P && \text{P/D} \\ &\geq |M| \\ &\geq \sum_{a\in A} y_a + \sum_{b\in B} y_b && \text{lemma 1} \\ &\geq (1-\varepsilon)\, opt_D && \text{lemma 2} \\ &\geq (1-\varepsilon)\, opt_{LP} \end{aligned}$$

Lemma 2: $y/(1-\varepsilon)$ will be a dual feasible solution

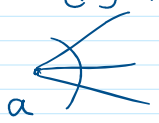
proof: we show $\forall\, ab\in E$: $y_a + y_b \geq 1-\varepsilon$

- if a is not matched: $\longrightarrow$ All its neighbors b have $y_b = 1$
  $\longrightarrow$ $y_a + y_b = 1$
- if $ab \in M$: $y_a = 1 - y_b \longrightarrow y_a + y_b = 1$
- if $ac \in M$ for some other node c:
  when ac was chosen, y_c was smallest $\longrightarrow y_c \leq y_b$ at the time, so after ac was added y_c got $+\varepsilon$; so $y_c \leq y_b + \varepsilon$ at that time.
  Since c remained assigned to a, y_c does nf increase, y_b can only increase $\longrightarrow$
  at the end: $y_c \leq y_b + \varepsilon$ but $\underbrace{y_a + y_c = 1}_{\text{by Fact 1}}$

  $\longrightarrow$ $y_a + y_b \geq 1-\varepsilon$.

Time Complexity: Each iteration of while loop the y_b value of some $b \in B$ increases by ε, and at most n nodes $\longrightarrow O(n/\varepsilon)$ iterations.

Add/delete to U takes $O(1)$ and $O(n)$ to check y_b for neighbours of a to find $\min y_b$ $\longrightarrow$ total time $O(n^2/\varepsilon) \rightsquigarrow O(m/\varepsilon)$



Better time: For each $a \in A$ be keep a list $D(a)$ of size $\deg(a)$ and a threshold value $d(a)$ initially $=0$.

Initially each $d(a)$ has all neighbours of a and to find $\min_{b \in N(a)} y_b$, we check nodes in $D(a)$ if $y_b > d(a)$ remove b from $D(a)$; else return b; when $D(a)$ becomes empty we increase threshold $d(a)$ by ε and add back all neighbours of a to $D(a)$.

Note that each time we pick $b \in D(a)$ it has the smallest y_b value among $N(a)$.

Also, $D(a)$ resets to $N(a)$ at most $\frac{1}{\varepsilon}$ times

$\longrightarrow$ time for each node $a \in A$: $O(\deg(a)/\varepsilon)$

$\longrightarrow$ total time of algorithm: $O(m/\varepsilon)$.

Change to Exact matching Algorithm

if we choose $\varepsilon < \frac{1}{n}$, say $\varepsilon = \frac{1}{n+1}$, then we find a matching M with $|M| \geq (1 - \frac{1}{n+1})\,opt$

Since opt $\leq n$. Since opt is $> \text{opt}-1$

always integer this implies $|M| = \text{opt}$ and time complexity is $O(mn)$.

Faster exact Algorithm:

- phase 1: run the $(1-\varepsilon)$-approx alg. with $\varepsilon = 1/\sqrt{n}$ to obtain a matching of size $(1-\frac{1}{\sqrt{n}})\,\text{opt} \geq \text{opt} - \sqrt{n}$ in time $O(m\sqrt{n})$.
- phase 2: run augumenting path algorithm at most $\sqrt{n}$ times to find a max matching in time $O(m\sqrt{n})$.