## Bipartite Matching \& the Hungarian Method

Last Revised: 7/10/06

These notes follow formulation developed by Subhash Suri in http://www.cs.ucsb.edu/suri/cs230/Matching.pdf

We previously saw how to use the Ford-Fulkerson MaxFlow algorithm to find Maximum-Size matchings in bipartite graphs. In this section we discuss how to find Maximum-Weight matchings in bipartite graphs, a situation in which Max-Flow is no longer applicable.

The $O\left(|V|^{3}\right)$ algorithm presented is the Hungarian AIgorithm due to Kuhn \& Munkres.

- Review of Max-Bipartite Matching

Earlier seen in Max-Flow section

- Augmenting Paths
- Feasible Labelings and Equality Graphs
- The Hungarian Algorithm for Max-Weighted Bipartite Matching


## Application: Max Bipartite Matching

A graph $G=(V, E)$ is bipartite if there exists partition $V=X \cup Y$ with $X \cap Y=\emptyset$ and $E \subseteq X \times Y$.

A Matching is a subset $M \subseteq E$ such that $\forall v \in V$ at most one edge in $M$ is incident upon $v$.

The size of a matching is $|M|$, the number of edges in M.

A Maximum Matching is matching $M$ such that every other matching $M^{\prime}$ satisfies $\left|M^{\prime}\right| \leq M$.

Problem: Given bipartite graph $G$, find a maximum matching.

A bipartite graph with 2 matchings


We now consider Weighted bipartite graphs. These are graphs in which each edge $(i, j)$ has a weight, or value, $w(i, j)$. The weight of matching $M$ is the sum of the weights of edges in $M, w(M)=\sum_{e \in M} w(e)$.

Problem: Given bipartite weighted graph $G$, find a maximum weight matching.


Note that, without loss of generality, by adding edges of weight 0 , we may assume that $G$ is a complete weighted graph.

## Alternating Paths:



- Let $M$ be a matching of $G$.
- Vertex $v$ is matched if it is endpoint of edge in $M$; otherwise $v$ is free
$Y_{2}, Y_{3}, Y_{4}, Y_{6}, X_{2}, X_{4}, X_{5}, X_{6}$ are matched, other vertices are free.
- A path is alternating if its edges alternate between $M$ and $E-M$. $Y_{1}, X_{2}, Y_{2}, X_{4}, Y_{4}, X_{5}, Y_{3}, X_{3}$ is alternating
- An alternating path is augmenting if both endpoints are free.
- Augmenting path has one less edge in $M$ than in $E-M$; replacing the $M$ edges by the $E-M$ ones increments size of the matching.


## Alternating Trees:



An alternating tree is a tree rooted at some free vertex $v$ in which every path is an alternating path.

Note: The diagram assumes a complete bipartite graph; matching $M$ is the red edges. Root is $Y_{5}$.

## The Assignment Problem:

Let $G$ be a (complete) weighted bipartite graph.

The Assignment problem is to find a max-weight matching in $G$.

A Perfect Matching is an $M$ in which every vertex is adjacent to some edge in $M$.

A max-weight matching is perfect.

Max-Flow reduction dosn't work in presence of weights. The algorithm we will see is called the Hungarian AIgorithm.

## Feasible Labelings \& Equality Graphs




Equality Graph $G_{\ell}$

- A vetex labeling is a function $\ell: V \rightarrow \mathcal{R}$
- A feasible labeling is one such that

$$
\ell(x)+\ell(y) \geq w(x, y), \quad \forall x \in X, y \in Y
$$

- the Equality Graph (with respect to $\ell$ ) is

$$
G=\left(V, E_{\ell}\right) \text { where }
$$

$$
E_{\ell}=\{(x, y): \ell(x)+\ell(y)=w(x, y)\}
$$



A feasible labeling $\ell$


Equality Graph $G_{\ell}$

Theorem: If $\ell$ is feasible and $M$ is a Perfect matching in $E_{\ell}$ then $M$ is a max-weight matching.

## Proof:

Denote edge $e \in E$ by $e=\left(e_{x}, e_{y}\right)$.
Let $M^{\prime}$ be any PM in $G$ (not necessarily in in $E_{\ell}$ ). Since every $v \in V$ is covered exactly once by $M$ we have
$w\left(M^{\prime}\right)=\sum_{e \in M^{\prime}} w(e) \leq \sum_{e \in M^{\prime}}\left(\ell\left(e_{x}\right)+\ell\left(e_{y}\right)\right)=\sum_{v \in V} \ell(v)$
so $\sum_{v \in V} \ell(v)$ is an upper-bound on the cost of any perfect matching.

Now let $M$ be a PM in $E_{\ell}$. Then

$$
w(M)=\sum_{e \in M} w(e)=\sum_{v \in V} \ell(v) .
$$

So $w\left(M^{\prime}\right) \leq w(M)$ and $M$ is optimal.


A feasible labeling $\ell$


Equality Graph $G_{\ell}$

Theorem[Kuhn-Munkres]: If $\ell$ is feasible and $M$ is a Perfect matching in $E_{\ell}$ then $M$ is a max-weight matching.

The KM theorem transforms the problem from an optimization problem of finding a max-weight matching into a combinatorial one of finding a perfect matching. It combinatorializes the weights. This is a classic technique in combinatorial optimization.

Notice that the proof of the KM theorem says that for any matching $M$ and any feasible labeling $\ell$ we have

$$
w(M) \leq \sum_{v \in V} \ell(v) .
$$

This has very strong echos of the max-flow min-cut theorem.

Our algorithm will be to
Start with any feasible labeling $\ell$ and some matching $M \subseteq E_{\ell}$ maintaining an alternating tree $\mathcal{T} \subseteq E_{\ell}$.

While $M$ is not perfect repeat the following:

1. Find an augmenting path for $M$ in $E_{\ell}$; this increases size of $M$
Reset $\mathcal{T}$ to be one free vertex
2. If no augmenting path exists, improve $\ell$ to $\ell^{\prime}$ such that $M, \mathcal{T} \subset E_{\ell^{\prime}}$. Add one edge in $E_{\ell^{\prime}}$ to $\mathcal{T}$, keeping it an augmenting tree Go to 1 .

Note that in each step of the loop we will either be increasing the size of $M$ or $\mathcal{T}$ so this process must terminate.

Furthermore, when the process terminates, $M$ will be a perfect matching in $E_{\ell}$ for some feasible labeling $\ell$. So, by the Kuhn-Munkres theorem, $M$ will be a maxweight matching.

## Finding an Initial Feasible Labelling



Finding an initial feasible labeling is simple. Just use:

$$
\forall y \in Y, \ell(y)=0, \quad \forall x \in X, \ell(x)=\max _{y \in Y}\{w(x, y)\}
$$

With this labelling it is obvious that

$$
\forall x \in X, y \in Y, w(x, y) \leq \ell(x)+\ell(y)
$$

## Improving Labellings

Let $\ell$ be a feasible labeling.
Define neighbor of $u \in V$ and set $S \subseteq V$ to be
$N_{\ell}(u)=\left\{v:(u, v) \in E_{\ell},\right\}, \quad N_{\ell}(S)=\cup_{u \in S} N_{\ell}(u)$

Lemma: Let $S \subseteq X$ and $T=N_{\ell}(S) \neq Y$. Set

$$
\alpha_{\ell}=\min _{x \in S, y \notin T}\{\ell(x)+\ell(y)-w(x, y)\}
$$

and

$$
\ell^{\prime}(v)= \begin{cases}\ell(v)-\alpha_{\ell} & \text { if } v \in S \\ \ell(v)+\alpha_{\ell} & \text { if } v \in T \\ \ell(v) & \text { otherwise }\end{cases}
$$

Then $\ell^{\prime}$ is a feasible labeling and,
(i) If $(x, y) \in E_{\ell}$ for $x \in S, y \in T$ then $(x, y) \in E_{\ell^{\prime}}$;
(ii) If $(x, y) \in E_{\ell}$ for $x \notin S, y \notin T$ then $(x, y) \in E_{\ell^{\prime}}$;
(iii) For some $x \in S, y \notin T$
we have $(x, y) \notin E_{\ell}$ but $(x, y) \in E_{\ell^{\prime}}$

## The Hungarian Method

1. Generate initial labelling $\ell$ and matching $M$ in $E_{\ell}$.
2. If $M$ perfect, stop.

Otherwise pick free vertex $u \in X$.
Set $S=\{u\}, T=\emptyset$.
Note: $S \cup T$ will be vertices of alternating tree
3. If $N_{\ell}(S)=T$, update labels (forcing $N_{\ell}(S) \neq T$ )

$$
\alpha_{\ell}=\min _{s \in S, y \notin T}\{\ell(x)+\ell(y)-w(x, y)\}
$$

$\ell^{\prime}(v)= \begin{cases}\ell(v)-\alpha_{\ell} & \text { if } v \in S \\ \ell(v)+\alpha_{\ell} & \text { if } v \in T \\ \ell(v) & \text { otherwise }\end{cases}$
4. If $N_{\ell}(S) \neq T$, pick $y \in N_{\ell}(S)-T$.

- If $y$ free, $u \leadsto y$ is augmenting path. Augment $M$ and go to 2 .
- If $y$ matched, say to $z$, extend alternating tree: $S=S \cup\{z\}, T=T \cup\{y\}$. Go to 3 .


Original Graph


Eq Graph+Matching


Alternating Tree

- Initial Graph, trivial labelling and associated Equality Graph
- Initial matching: $\left(x_{3}, y_{1}\right),\left(x_{2}, y_{2}\right)$
- $S=\left\{x_{1}\right\}, T=\emptyset$.
- Since $N_{\ell}(S) \neq T$, do step 4. Choose $y_{2} \in N_{\ell}(S)-T$.
- $y_{2}$ is matched so grow tree by adding $\left(y_{2}, x_{2}\right)$, i.e., $S=\left\{x_{1}, x_{2}\right\}, T=\left\{y_{2}\right\}$.
- At this point $N_{\ell}(S)=T$, so goto 3 .


Original Graph


Old $E_{\ell}$ and $|M|$

new Eq Graph

- $S=\left\{x_{1}, x_{2}\right\}, T=\left\{y_{2}\right\}$ and $N_{\ell}(S)=T$
- Calculate $\alpha_{\ell}$

$$
\begin{aligned}
\alpha_{\ell} & =\min _{x \in S, y \notin T} \begin{cases}6+0-1, & \left(x_{1}, y_{1}\right) \\
6+0-0, & \left(x_{1}, y_{3}\right) \\
8+0-0, & \left(x_{2}, y_{1}\right) \\
8+0-6, & \left(x_{2}, y_{3}\right)\end{cases} \\
& =2
\end{aligned}
$$

- Reduce labels of $S$ by 2 ;

Increase labels of $T$ by 2 .

- Now $N_{\ell}(S)=\left\{y_{2}, y_{3}\right\} \neq\left\{y_{2}\right\}=T$.


Orig $E_{\ell}$ and $M$


New Alternating Tree


New $M$

- $S=\left\{x_{1}, x_{2}\right\}, N_{\ell}(S)=\left\{y_{2}, y_{3}\right\}, T=\left\{y_{2}\right\}$
- Choose $y_{3} \in N_{\ell}(S)-T$ and add it to $T$.
- $y_{3}$ is not matched in $M$ so we have just found an alternating path $x_{1}, y_{2}, x_{2}, y_{3}$ with two free endpoints. We can therefore augment $M$ to get a larger matching in the new equality graph. This matching is perfect, so it must be optimal.
- Note that matching $\left(x_{1}, y_{2}\right),\left(x_{2}, y_{3}\right),\left(x_{3}, y_{1}\right)$ has cost $6+6+4=16$ which is exactly the sum of the labels in our final feasible labelling.


## Correctness:

- We can always take the trivial $\ell$ and empty matching $M=\emptyset$ to start algorithm.
- If $N_{\ell}(S)=T$, we saw that we could always update labels to create a new feasible matching $\ell^{\prime}$. The lemma on page 13 guarantees that all edges in $S \times T$ and $\bar{S} \times \bar{T}$ that were in $E_{\ell}$ will be in $E_{\ell^{\prime}}$. In particular, this guarantees (why?) that the current $M$ remains in $E_{\ell^{\prime}}$ as does the alternating tree built so far.

Note: The lemma requires that $T \neq Y$ but this is trivially correct since $|T|=|S|-1$ so $|T|<|Y|$.

- If $N_{\ell}(S) \neq T$, we can, by definition, always augment alternating tree by choosing some $x \in S$ and $y \notin T$ such that $(x, y) \in E_{\ell}$. Note that at some point $y$ chosen must be free, in which case we augment $M$.
- So the algorithm always terminates.
- $M$ is a perfect matching in $E_{\ell}$ when the algorithm terminates
—it is optimal by Kuhn-Munkres theorem.


## Complexity

In each phase of algorithm, $|M|$ increases by 1 so there are at most $V$ phases. How much work needs to be done in each phase?

In implementation, $\forall y \notin T$ keep track of slack $_{y}=\min _{x \in S}\{\ell(x)+\ell(y)-w(x, y)\}$

- Initializing all slacks at beginning of phase takes $O(|V|)$ time.
- In step 4 we must update all slacks when vertex moves from $\bar{S}$ to $S$.
This takes $O(|V|)$ time; only $|V|$ vertices can be moved from $\bar{S}$ to $S$, giving $O\left(|V|^{2}\right)$ time per phase.
- In step 3, $\alpha_{\ell}=\min _{y \notin T}$ slack $_{y}$ and can therefore be calculated in $O(|V|)$ time from the slacks. This is done at most $|V|$ times per phase (why?) so only takes $O\left(|V|^{2}\right)$ time per phase.
After calculating $\alpha_{\ell}$ we must update all slacks. This can be done in $O(|V|)$ time by setting

$$
\forall y \notin T, \text { slack }_{y}=\text { slack }_{y}-\alpha_{\ell} .
$$

Since this is only done $O(|V|)$ times, total time per phase is $O\left(|V|^{2}\right)$.

There are $|V|$ phases and $O\left(|V|^{2}\right)$ work per phase so the total running time is $O\left(|V|^{3}\right)$.

