Weighted bipartite matchings

Bora Uçar

RO:MA, LIP, ENS Lyon, France

CR-08: Combinatorial scientific computing, November 2014
http://perso.ens-lyon.fr/bora.ucar/CR08/
Outline

1. Introduction

2. The Hungarian algorithm for the weighted matching problem

3. Weighted matching and linear solvers
1. Introduction

2. The Hungarian algorithm for the weighted matching problem

3. Weighted matching and linear solvers
We have seen the maximum cardinality matching on bipartite graphs, where we wanted to find a matching $\mathcal{M}$ of maximum cardinality, i.e., we maximized $|\mathcal{M}|$.

We have used to detect fully indecomposable parts of a matrix.

We now look at matchings that will be helpful in numerical aspects. Recall that in LU, we divide by pivots. Having bigger numbers there will help.
Consider bipartite graphs where each edge \((i, j)\) has a weight \(w(i, j)\). The weight of a matching \(\mathcal{M}\) is the sum of the weights in \(\mathcal{M}\), i.e.,

\[ w(\mathcal{M}) = \sum_{(i, j) \in \mathcal{M}} w(i, j). \]

We consider the maximum-weighted perfect matching problem (bipartite graphs with the two parts having the same number of vertices \(n\), where the maximum cardinality of a matching is \(n\)).
Outline

1. Introduction
2. The Hungarian algorithm for the weighted matching problem
3. Weighted matching and linear solvers
Maximum weighted matching

- A **vertex labeling** is a function associating weights to vertices \( \ell : V \rightarrow \mathbb{R} \).

- A labeling is feasible if

\[
\ell(x) + \ell(y) \geq w(x, y) \text{ for all edges } (x, y)
\]

- The **equality graph** with respect to \( \ell \) is defined as \( G_\ell = (V, E_\ell) \)

where

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

(some slides from Mordecai J. Golin)
Maximum weighted matching

A feasible labeling $\ell$

Equality Graph $G_\ell$
Kuhn-Munkres theorem

**Theorem:** If $\ell$ is feasible and $\mathcal{M}$ is a perfect matching in $E_\ell$, then $\mathcal{M}$ is a maximum-weighted matching.

**Proof**

\[\langle\langle\text{On the board....}\rangle\rangle\]

Two things:

- an upper bound on the weight of a matching;
- show that the weight of a perfect matching in $E_\ell$ matches to that bound.

\[\rangle\rangle\]
The algorithm will be

1. start with any feasible labeling $\ell$ and some matching $M$ in $E_\ell$
2. while $M$ is not a perfect matching
   - find an augmenting path for $M$ in $E_\ell$
   - if no augmenting path exists, improve $\ell$ to $\ell'$ such that we get closer to a perfect matching with $\ell'$

In each step, either the cardinality of $M$ increases or get more edges (incident on some matched vertices); hence the process must terminate.

At the end, the matching $M$ will be a perfect matching in $E_\ell$ for some feasible labeling $\ell$. By the theorem above, we will have a maximum-weighted matching.
Hungarian algorithm: Initial feasible labeling

Take $\ell(y) = 0$ for all $y \in Y$ and $\ell(x) = \max_{y \in Y} w(x, y)$ for all $x \in X$.

The labeling is feasible, $w(x, y) \leq \ell(x) + \ell(y)$.
Hungarian algorithm: Improving a labeling

Let $\ell$ be a feasible labeling. Define neighbor of $u \in V$ and set $S \subseteq V$ to be

$$N_\ell(u) = \{v : (u,v) \in E_\ell\}, \quad N_\ell(S) = \bigcup_{u \in S} N_\ell(u)$$

**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

$$\ell'(v) = \begin{cases} 
\ell(v) - \alpha_\ell & \text{if } v \in S \\
\ell(v) + \alpha_\ell & \text{if } v \in T \\
\ell(v) & \text{otherwise}
\end{cases}$$

Then $\ell'$ is a feasible labeling and

(i) If $(x,y) \in E_\ell$ for $x \in S, y \in T$ then $(x,y) \in E_{\ell'}$.
(ii) If $(x,y) \in E_\ell$ for $x \notin S, y \notin T$ then $(x,y) \in E_{\ell'}$.
(iii) There is some edge $(x,y) \in E_{\ell'}$ for $x \in S, y \notin T$. 
Hungarian algorithm: Improving a labeling

Proof of the lemma:

⟨⟨ On the board ⟩⟩

Note: an edge of the form \((\tilde{S}, T)\) can be dropped from the equality graph.
Hungarian algorithm

1. Generate initial labelling $\ell$ and matching $M$ in $E_\ell$.

2. If $M$ perfect, stop.
   Otherwise pick free vertex $u \in X$.
   Set $S = \{u\}$, $T = \emptyset$.

3. If $N_\ell(S) = T$, update labels (forcing $N_\ell(S) \neq T$)
   \[ \alpha_\ell = \min_{s \in S, y \not\in T} \{ \ell(x) + \ell(y) - w(x, y) \} \]
   \[ \ell'(v) = \begin{cases} 
   \ell(v) - \alpha_\ell & \text{if } v \in S \\
   \ell(v) + \alpha_\ell & \text{if } v \in T \\
   \ell(v) & \text{otherwise} 
   \end{cases} \]

4. If $N_\ell(S) \neq T$, pick $y \in N_\ell(S) - T$.
   - If $y$ free, $u - y$ 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.
Hungarian algorithm: Example (1)

Initial Graph, trivial labelling and associated Equality Graph

Initial matching: \((x_3, y_1), (x_2, y_2)\)

\[ S = \{x_1\}, \quad T = \emptyset. \]

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 \((y_2, x_2)\), i.e., \(S = \{x_1, x_2\}, T = \{y_2\}\).

At this point \(N_\ell(S) = T\), so goto 3.
Hungarian algorithm: Example (2)

Original Graph

\[ S = \{x_1, x_2\}, T = \{y_2\} \]

and \( N_\ell(S) = T \)

Calculate \( \alpha_\ell \)

\[
\alpha_\ell = \min_{x \in S, y \notin T} \left\{ 6 + 0 - 1, (x_1, y_1) \right\} \quad 6 + 0 - 0, (x_1, y_3) \quad 8 + 0 - 0, (x_2, y_1) \quad 8 + 0 - 6, (x_2, y_3) \]

\[ = 2 \]

Old \( E_\ell \) and \( |M| \)

new Eq Graph

Reduce labels of \( S \) by 2;
Increase labels of \( T \) by 2.

Now \( N_\ell(S) = \{y_2, y_3\} \neq \{y_2\} = T \).
Hungarian algorithm: Example (3)

\[ S = \{x_1, x_2\}, \quad N_\ell(S) = \{y_2, y_3\}, \quad T = \{y_2\} \]

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.

The matching (in green) has cost \( 6 + 6 + 4 = 16 \) which is equal to the sum of the final labels.
Outline

1. Introduction

2. The Hungarian algorithm for the weighted matching problem

3. Weighted matching and linear solvers
Given an $n \times n$ matrix $A$, find a permutation $M$ such that the diagonal product of the permuted matrix, $\prod \text{diag}(AM)$, is maximum (in magnitude) among all permutations. Assume $a_{ij} \geq 0$ and there is at least one nonzero product diagonal (full structural rank).

How to solve it using the maximum weighted matching?
Maximum product matching

Given an $n \times n$ matrix $A$, find a permutation $M$ such that the diagonal product of the permuted matrix, $\prod \text{diag}(AM)$, is maximum (in magnitude).

The maximization of the product of entries can be translated into the maximization of the sum of the entries by defining a matrix $C = (c_{ij})_{n \times n}$ with

$$c_{ij} = \begin{cases} \log |a_{ij}| & \text{if } a_{ij} \neq 0 \\ -\infty & \text{otherwise} \end{cases}$$

In other words, we look for a maximum weighted perfect matching in the bipartite graph of $C$. 
Maximum product matching

There is more to a maximum weighted perfect matching on \( C \).

Consider the labels at the end of the Hungarian algorithm and recall the conditions we met with the labels

\[
\begin{align*}
\ell(i) + \ell(j) &\geq c_{ij} \quad \text{for } (i, j) \in E \\
\ell(i) + \ell(j) &= c_{ij} \quad \text{for } (i, j) \in M
\end{align*}
\]

If we translate these equations to the original matrix, we obtain

\[
\begin{align*}
e^{\ell(i)}e^{\ell(j)} &\geq a_{ij} \quad \text{for } (i, j) \in E \\
e^{\ell(i)}e^{\ell(j)} &= a_{ij} \quad \text{for } (i, j) \in M
\end{align*}
\]

In other words, if we scale the permuted matrix \( AM \) on the left by \( D_R = \text{diag}(1/e^{\ell(i)})_{i=1,...,n} \) and on the right by \( D_C = \text{diag}(1/e^{\ell(j)})_{j=1,...,n} \), we obtain a matrix with 1 on the main diag, where all other entries are \( \leq 1 \). This is an effective heuristic for reducing the need for numerical pivoting.
Maximum weighted matching

Structural + numerical problem:

- **Maximum weighted transversal:**
  find a permutation $P$ of the columns of $A$ such that the product (or another metric: min, sum ...) of the diagonal entries of $AP$ is maximum.
Impact of maximum weighted matching algorithms

- **Influence of maximum weighted matching on the performance with MUMPS**

| Matrix       | Symmetry | $|LU|$ $(10^6)$ | Flops $(10^9)$ | Backwd Error |
|--------------|----------|--------------|---------------|--------------|
| TWOTONE      | OFF      | 28           | 235           | 1221         | $10^{-6}$    |
|              | ON       | 43           | 22            | 29           | $10^{-12}$   |
| FIDAPM11     | OFF      | 100          | 16            | 10           | $10^{-10}$   |
|              | ON       | 46           | 28            | 29           | $10^{-11}$   |

- On very unsymmetric matrices: reduce flops, factor size and memory used.
- In general improve accuracy, and reduce number of iterative refinements.
- Limit numerical pivoting / improve reliability of memory estimates.
Combine maximum transversal and fill-in reduction

- Consider the \( LU \) factorization \( A = LU \) of an unsymmetric matrix.

- Compute the column permutation \( Q \) leading to a maximum numerical transversal of \( A \). \( AQ \) has large (we have seen the max product; but max sum; or max/min are also possible) numerical entries on the diagonal.

- Find fill-reducing ordering of \( AQ \) preserving the diagonal entries. Equivalent to finding symmetric permutation \( P \) such that the factorization of \( PAQP^T \) has reduced fill-in.
Homework Question: Let $A$ be an $n \times n$ nonnegative matrix (that is $a_{ij} \geq 0$) that has at least one perfect matching. Design an algorithm that finds a permutation $M$ such that the minimum element in the diagonal of the permuted matrix, $\min(\text{diag}(AM))$, is maximum among all permutations.