Finding weighted matchings for singular sparse symmetric matrices

Jennifer Scott
STFC Rutherford Appleton Laboratory
in collaboration with Jonathan Hogg

Sparse Days at CERFACS, 25th June 2012

Outline of talk

- Introduction and motivation
- Finding weighted matchings
  - review of how this is done
  - how to get optimal result in singular case
- Some results and comments

Basic problem

- Maximum matching of a sparse matrix A that maximizes product of matched entries can be used to increase the speed and reliability of sparse linear algebra operations (MC64).
- One popular method of solution: transform to an assignment problem and use sparse variant of Hungarian algorithm.
- If A structurally rank deficient, chooses set $\mathcal{I} \times \mathcal{J}$ of rows and columns such that restriction of A to $\mathcal{I} \times \mathcal{J}$ is non-singular but, in general, chosen $\mathcal{I} \times \mathcal{J}$ is sub-optimal.

Aim: modify approach to obtain optimal $\mathcal{I} \times \mathcal{J}$.

What's a matching?

Let $A = \{a_{ij}\}$ be a $n \times n$ sparse matrix.

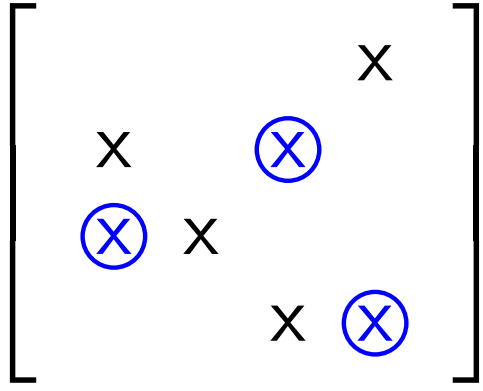
Let $\mathcal{G}_A = (V_r \cup V_c, E)$ be bipartite graph.

Node sets $V_r = \{1, 2, .., n\}$ and $V_c = \{1, 2, .., n\}$ correspond to rows and columns of A.

An edge (i, j) belongs to E if and only if $a_{ij} \neq 0$.

$\mathcal{M} \subseteq E$ is a matching if no two edges in $\mathcal{M}$ are incident to the same node.
i.e. set of nonzero entries with no two in the same row or column.

Example



A node is matched if there is an edge in the matching incident on the node and is unmatched otherwise.

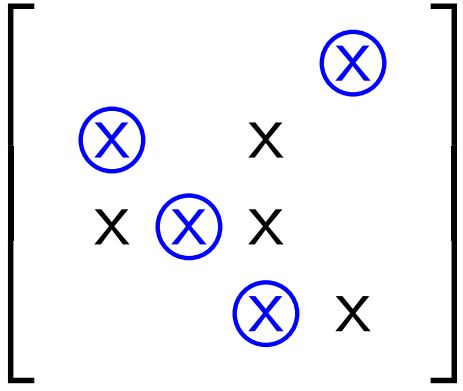
eg row 1 and column 2 are unmatched nodes.

What's a maximum matching?

The cardinality of a matching is the number of edges in it.

A maximum matching is a matching of maximum cardinality (n if A structurally non-singular).

Same example as before but now maximum matching.



Maximum product matching problem

Core problem: given A, find a matching of rows to columns that maximizes the product of the matched entries.

i.e., find permutation $\sigma = \{\sigma(i)\}$ that solves maximum product matching problem

$$\max_{\sigma} \prod_{i=1}^{n} |a_{i\sigma(i)}|.$$

Why is this important?

Allows large entries to be placed on diagonal.

Matrix can then be scaled so that diagonal entries have modulus 1 and off-diagonal entries have modulus ≤ 1.

Greatly improves numerical stability of subsequent *LU* factorization.

Also significant benefits for iterative methods/preconditioners.

Hence widely used and, in particular, implementation provided by HSL package `MC64` of Duff and Koster.

What if A is rank deficient?

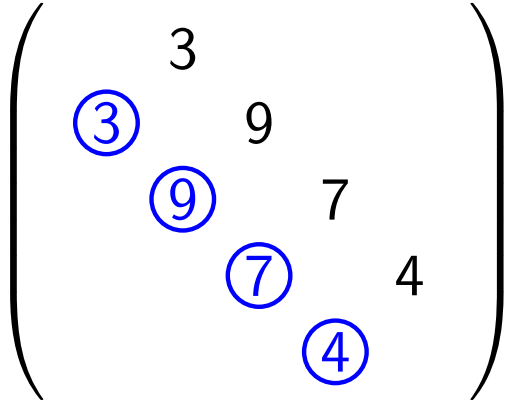
If A is structurally rank deficient with rank $r < n$, problem is ill-posed because the objective is 0 for all possible σ.

In this case, want a matching on $\mathcal{I} \times \mathcal{J}$ (where $\mathcal{I} \in V_r$ and $\mathcal{J} \in V_c$) of cardinality r that solves

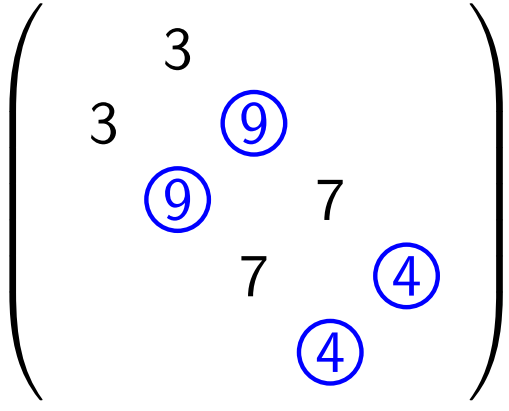
$$\max_{\mathcal{I}:|\mathcal{I}|=r} \max_{\sigma} \prod_{i \in \mathcal{I}} |a_{i\sigma(i)}|.$$

Duff and Koster approach fails to do this. It returns a matching of cardinality r that is optimal on a submatrix of A.

Example of sub-optimality of Duff and Koster



Duff and Koster matching



Optimal matching

What if A is symmetric?

A symmetric matrix need not have a symmetric matching. eg,

$$\begin{bmatrix} & \textcircled{1} & 2 \\ 1 & & \textcircled{3} \\ \textcircled{2} & 3 & \end{bmatrix}$$

But generally important when scaling a symmetric matrix that symmetry is preserved.

MC64 does not normally preserve symmetry.

Duff and Pralet show how to use MC64 to symmetrically scale A.

Specifically, let $S_r = \{s_{r_i}\}$ and $S_c = \{s_{c_i}\}$ be the row and column scalings from MC64. Set

$$S = \{\sqrt{s_{r_i} s_{c_i}}\}$$

and symmetrically scale SAS.

Entries in scaled matrix have absolute value ≤ 1 and entries in matching have absolute value 1.

What if A is symmetric and rank deficient?

Duff and Pralet prove if A symmetric and there is a maximum matching of A on $\mathcal{I} \times \mathcal{J}$ then $A_{\mathcal{I}\times\mathcal{I}}$ (restriction of A to $\mathcal{I} \times \mathcal{I}$) is structurally non-singular.

Thus there is a maximum matching for $A_{\mathcal{I}\times\mathcal{I}}$ of cardinality r and this can be used to obtain a matching for A.

But approach of D&P may lead to a sub-optimal matching (i.e., the choice of $\mathcal{I}$ may be sub-optimal).

Our aim: develop modified approach that gives optimal solution.

Duff and Koster algorithm (general A)

Transform maximum product matching problem to assignment problem that can be solved with minimal modifications to standard methods.

Let $c_j = \max_i |a_{ij}|$ be maximum absolute value in column j.

Preprocess A to $\hat{A}$ with

$$\hat{a}_{ij} = \begin{cases} \log c_j - \log |a_{ij}|, & \text{for } a_{ij} \neq 0, \\ \infty & \text{otherwise.} \end{cases}$$

The linear sum assignment problem

$$\min \sum_{i=1}^{n} \hat{a}_{i\sigma(i)} \tag{1}$$

has same optimal permutation as maximum product matching problem.

Duff and Koster algorithm (general A)

Further, if there exists u_i and v_j with

$$\hat{a}_{ij} - u_i - v_j \begin{cases} = 0, & \text{for } j = \sigma(i), \\ \geq 0, & \text{otherwise,} \end{cases}$$

then u_i and v_j are optimal dual variables for (1).

Dual solution can be used to scale A by setting

$$s_i^r = exp(u_i),$$
$$s_j^c = exp(v_j - \log c_j).$$

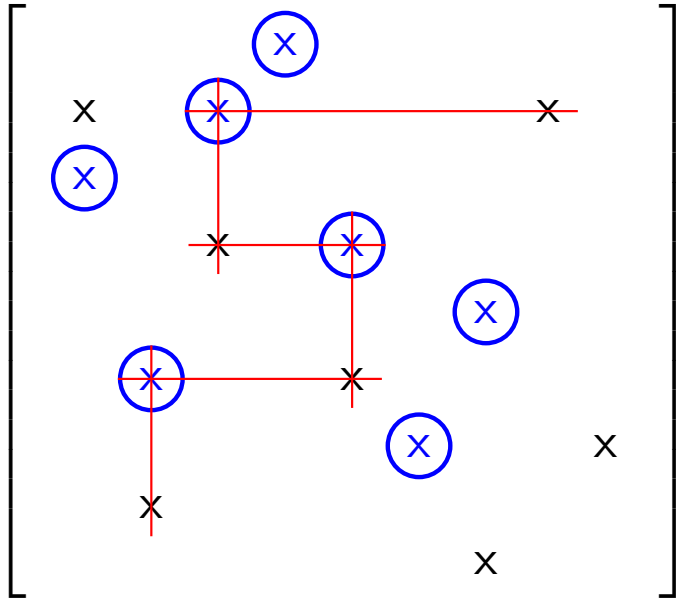
Scaled and permuted matrix has diagonal entries that are 1 in absolute value and off-diagonal entries that are ≤ 1.

What's an augmenting path?

$\mathcal{P}$ is an $\mathcal{M}$-alternating path if edges of $\mathcal{P}$ are alternately in $\mathcal{M}$ and not in $\mathcal{M}$.

An $\mathcal{M}$-alternating path is an $\mathcal{M}$-augmenting path if it connects an unmatched column with an unmatched row.

eg. 9×9 matrix, $\mathcal{M} = 7$, alternating path from $(2, 8)$ to $(8, 2)$.



Duff and Koster algorithm (general A)

Start: Preprocessed matrix $\hat{A}$, an initial matching σ and dual variables u, v that are optimal on $\mathcal{I} \times \mathcal{J}$.

At each step, extend $\mathcal{I} \times \mathcal{J}$ by selecting an unmatched column j and then finding shortest σ-augmenting path $\mathcal{P}$ from j to an unmatched row i.

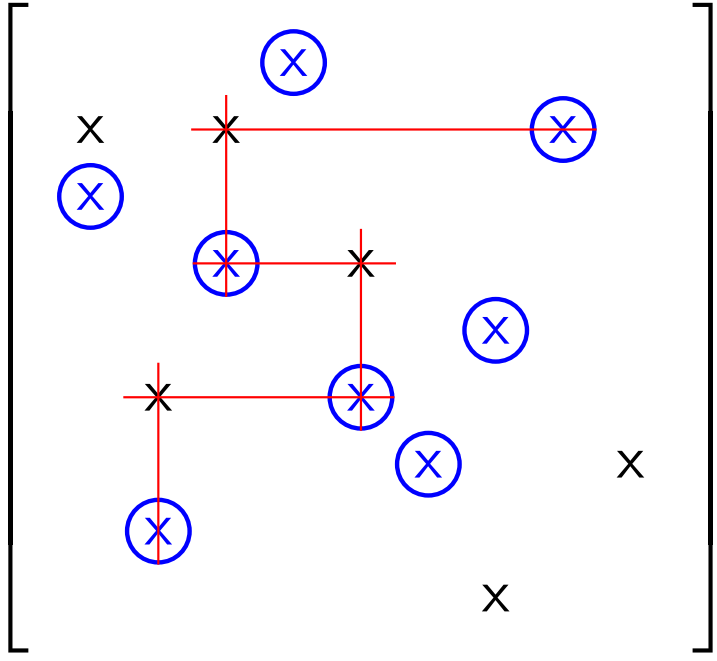
i.e., choose entry $\hat{a}_{tj}$ in unmatched column j that belongs to a matched row t; next entry in the path is $\hat{a}_{t\sigma(t)}$. The matched column $\sigma(t)$ is then searched and an unmatched entry chosen.

Once shortest $\mathcal{P}$ is found, σ is augmented along it, (all unmatched entries on $\mathcal{P}$ become matched entries, while previously matched entries become unmatched).

New matching is optimal on $\tilde{\mathcal{I}} \times \tilde{\mathcal{J}}$ with $\tilde{\mathcal{I}} = \mathcal{I} \cup \{i\}$ and $\tilde{\mathcal{J}} = \mathcal{J} \cup \{j\}$.

Extending a matching

After augmenting, $\mathcal{M} = 8$.



Dual variables u and v must be updated for new matching.

Let *lsap* be length of shortest augmenting path and $l(k)$ be length of shortest alternating path from j to column k.

For each row t for which $l(\sigma(t)) < lsap$,

$$u_t \leftarrow u_t + l(\sigma(t)) - lsap$$

and then for each $t \in \tilde{\mathcal{I}}$

$$v_{\sigma(t)} = \hat{a}_{t\sigma(t)} - u_t.$$

Duff and Pralet algorithm for symmetric A

1. Preprocess to A to obtain $\hat{A}$.
2. Determine initial matching σ and dual variables u, v that are optimal on $\mathcal{I}_{in} \times \mathcal{J}_{in}$.
3. Apply Duff and Koster to $\hat{A}$ to obtain maximum matching on $\mathcal{I} \times \mathcal{J}$.
4. `if` $(|\mathcal{I}| = n)$ `then`
   Compute scaling factors.
5. `else` *! structurally singular case*
   Apply Duff and Koster to $A_{\mathcal{I}\times\mathcal{I}}$.
   Use maximum matching for $A_{\mathcal{I}\times\mathcal{I}}$ to compute matching for A.
   Compute scaling factors.

   `end if`

Symmetric preprocessing

Duff and Pralet use unsymmetric preprocessing

$$\hat{a}_{ij} = \log c_j - \log |a_{ij}|, \text{ with } c_j = \max_i |a_{ij}|$$

If A is symmetric, $\hat{A}$ is generally unsymmetric.

Undesirable in the rank deficient case eg consider 3×3 example where maximum product matching is circled

$$\begin{bmatrix} 1.0 & 10^3 & \textcircled{10^9} \\ 10^3 & & \\ \textcircled{10^9} & & \end{bmatrix}.$$

Preprocessing (using $\log_{10}$ for convenience) gives

$$\begin{bmatrix} 9 & \textcircled{0} & 0 \\ 6 & & \\ \textcircled{0} & & \end{bmatrix}.$$

In this case, impossible to distinguish columns 2 and 3 and columns 1 and 2 are matched to rows 3 and 1, respectively.

Symmetric preprocessing

For symmetric matrices, to retain symmetry, we propose symmetric preprocessing

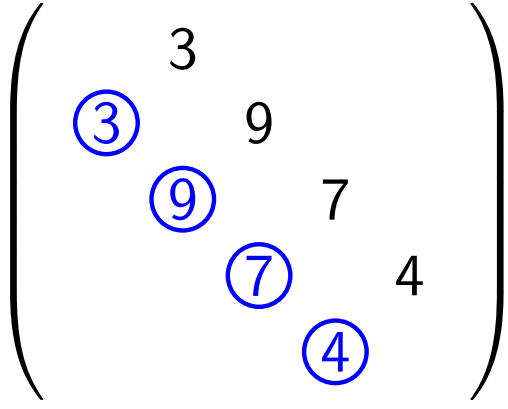
$$\hat{a}_{ij} = (\log c_j + \log r_i)/2 - \log |a_{ij}|, \qquad (2)$$

where $r_i = \max_j |a_{ij}|$ is the maximum absolute value in row i.
For simple example, now get desired matching

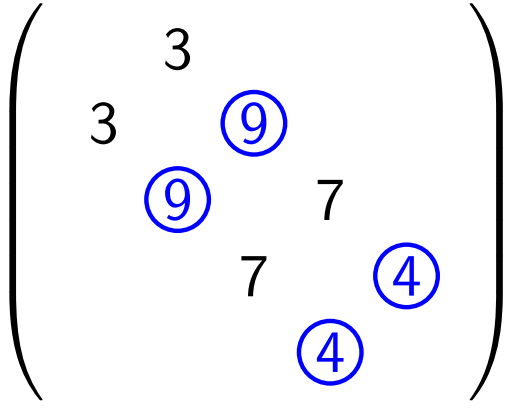
$$\begin{bmatrix} 9 & 3 & \textcircled{0} \\ 3 & & \\ \textcircled{0} & & \end{bmatrix}$$

But this will not always be the case so need to do something else.

Recall sub-optimality of Duff and Koster



Duff and Koster matching



Optimal matching

Modified algorithm for symmetric rank deficient matrices

Apply Duff and Koster to modified matrix

$$\begin{pmatrix} \epsilon & 3 & \epsilon & \epsilon & \epsilon \\ 3 & \epsilon & 9 & \epsilon & \epsilon \\ \epsilon & 9 & \epsilon & 7 & \epsilon \\ \epsilon & \epsilon & 7 & \epsilon & 4 \\ \epsilon & \epsilon & \epsilon & 4 & \epsilon \end{pmatrix}$$

in which zeros replaced by $\epsilon \neq 0$, a number so small that it will never be chosen as part of the matching if there is an alternative.

Rank-deficiency is apparent in that optimal matching contains ϵ.

An optimal matching for original matrix obtained by restricting matching to those rows and columns that are not matched on an ϵ.

Not practical for real problems but underlying approach can be used provided ϵ entries handled implicitly.

Denote modified matrix by $A(\epsilon)$.
Preprocessing transforms each ϵ entry to

$$\hat{a}_{ij}(\epsilon) = (\log c_j + \log r_i)/2 + \alpha,$$

where α is very large. Do not store but compute on fly.

Call such entries α entries.

In structurally non-singular case, an α entry is never chosen (too large).

In rank deficient case, do not include α entries in the matching but when α entry occurs in the path, augment along the path excluding the α value.

Size of the matching will not increase but the optimal value will be improved.

i.e., once algorithm starts considering α entries, $|\mathcal{I}|$ and $|\mathcal{J}|$ remain constant, with one column (or row) swapped for another at each step.

We skip implementation details but key points are:

- Duff and Koster is implemented columnwise.
- We work alternately columnwise to substitute a column k for a column j in $\mathcal{J}$ and then rowwise to substitute a row l for a row i in $\mathcal{I}$.
- Equivalent to working alternately with A and then A^T.
- The objective function value improves on each step so terminates with the optimal in a finite number of steps (we found single pass on A and A^T to be sufficient).
- Necessary to take action to prevent cycling (swap k and j and then j with k).

Note: only use A and A^T in the structurally singular case.

In non-singular case, reduces to Duff and Koster with symmetric preprocessing and the symmetric scaling factors of Duff and Pralet.

Test problems ($n - r$ is rank deficiency)

Name	n	$n-r$	(%)	nz
GHS_indef/dtoc	24 993	4 999	(20.0)	69 972
Newman/astro-ph	16 706	990	(5.92)	242 502
Newman/cond-mat-2005	40 421	2 453	(6.07)	351 382
Newman/netscience	1 589	165	(10.4)	5 484
Pajek/Reuters911	13 332	2 647	(19.9)	296 076
SNAP/Oregon-1	11 492	8 170	(71.1)	46 818
Newman/as-22july06	22 963	16 362	(71.2)	96 872
AG-Monien/diag	2 559	169	(6.60)	8 184
DIMACS10/kron_g500-logn16	65 536	24 076	(36.7)	4 912 469

Reported statistic is

$$\log \prod_{i \in \mathcal{I}} |a_{i\sigma(i)}|$$

The better the matching, the greater this value is.

Unsymmetric matching

DK = Duff and Koster (`MC64`)
HS = Hogg and Scott (achieves optimal value)

Name	DK	HS
GHS_indef/dtoc	-42 577	0.00
Newman/astro-ph	-15 253	-15 090
Newman/cond-mat-2005	-29 695	-28 673
Newman/netscience	-1 072	-1 059
Pajek/Reuters911	2 304	4 010
SNAP/Oregon-1	-2 828	-1 884
Newman/as-22july06	-5 523	-3 741
AG-Monien/diag	-1 268	-952
DIMACS10/kron_g500-logn16	9	606

Symmetric matching

DP = Duff and Pralet; HS = Hogg and Scott

Name	DP	HS
GHS_indef/dtoc	-85 155	0.00
Newman/astro-ph	-15 231	-15 090
Newman/cond-mat-2005	-29 392	-2 8673
Newman/netscience	-1 066	-1 059
Pajek/Reuters911	2 497	4 010
SNAP/Oregon-1	-2 725	-1 884
Newman/as-22july06	-5 354	-3 741
AG-Monien/diag	-1 037	-952
DIMACS10/kron_g500-logn16	14	606

Note: DK and DP sensitive to initial ordering of A.

Good news

We have improved on the matching (get optimal results).

Bad news

Can be expensive (typically, HS costs twice DP but cost may be even more).

Is the extra cost worth it?

Indefinite sparse matrix factorizations: want to pick a pivot sequence that can be used without modification by factorization phase (with numerical pivoting).

Aim: permute large off-diagonal entry a_{ij} close to the diagonal so that 2×2 block

$$\begin{pmatrix} a_{ii} & a_{ij} \\ a_{ij} & a_{jj} \end{pmatrix}$$

is potentially good 2×2 candidate pivot.

Duff and Gilbert (2002) first observed cycle structure of permutation σ associated with an unsymmetric matching $\mathcal{M}$ can be exploited to obtain such a permutation.

Idea used in (eg) MUMPS, PARDISO, WSMP.

In non-singular case, can work well (can reduce number of delayed pivots and improve efficiency of factorization).

In singular case, had hoped better matchings would improve performance.

But this does not happen.

Computation of pivot order assumes diagonal present and so, although candidate 2×2 pivots picked by matching are good pivots, remaining 1×1 candidates from unmatched rows may not be and may be placed ahead of 2×2 pivots and lead to delays.

Note: Reducing pivoting threshold parameter does not help.

Observe: if $A_{\mathcal{I}\times\mathcal{I}}$ is numerically non-singular, we can set scaling factors

$$s_i = 0 \text{ for } i \notin \mathcal{I}$$

Example: GHS_indef/aug2d. $n = 29008$, $n - r = 9800$.

	$s_i \neq 0$	$s_i = 0$
No. entries in factors	$1.26 * 10^7$	$8.20 * 10^5$
No. flops	$2.36 * 10^{10}$	$5.36 * 10^7$
Factorization time	4.47	0.17

But if $A_{\mathcal{I}\times\mathcal{I}}$ is numerically singular, this approach will not work (will compute unacceptable solution).

We need more sophisticated approach during the factorization to swap rows in and out of $\mathcal{I}$.

Probably not worthwhile if r is small but where r is large, could be very advantageous.

Concluding remarks

We have developed an algorithm that computes the optimal matching in the structurally singular case.

So far, not been able to use it to improve sparse indefinite factorization as requires more fundamental reworking of factorization.

Are there other applications where optimal matching may be beneficial? Possibly unsymmetric rectangular case?

Thank you!