Sparse matrices

Guangning (Gary) Tan

tgnteach@gmail.com

CAS708/CSE700

McMaster University

September 29, 2015

Contents

This PDF can be accessed at

tgn3000.com ⇒ Teaching ⇒ CAS708/CSE700 Week 4

Why sparse matrices?

- Solve $A\mathbf{x} = \mathbf{b}$

$$A = \begin{bmatrix} 4 & 2 & & & & & \\ 2 & 5 & 2 & & & & \\ & 2 & -8 & -5 & & & \\ & & -5 & 9 & -3 & & \\ & & & -3 & -5 & 1 & \\ & & & & 1 & 5 & 3 \\ & & & & & 3 & 4 \end{bmatrix}$$

- Blank denotes 0

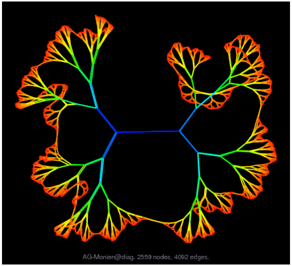
Why sparse matrices? (cont.)

- Save storage
- *n*: problem size

  τ: number of nontrivial (usually nonzero) entries
- Sparse algorithms are more efficient for sparse problems
  - Maintain sparsity pattern
  - Process only the nonzeros
  - E.g., LU decomposition for banded matrices
  - May contain parallelism
- May behave poorly on dense problems

The University of Florida Sparse Matrix Collection

- https://www.cise.ufl.edu/research/sparse/matrices/
- Download matrices using its Matlab's interface `UFget`
- Timothy A. Davis
  - *Direct Methods for Sparse Linear Systems.* 2006
  - *CSparse 3.1.4*, software in C code, 2014
  - http://faculty.cse.tamu.edu/davis/research.html



- Yifan Hu http://yifanhu.net/GALLERY/GRAPHS/index.html
- Matrix: AG-Monien/diag

Storing a sparse matrix

- Triplet form `(i,j,Aij)`

$$A = \begin{bmatrix} 0 & 2 & 5 & -2 \\ 1 & 0 & 0 & -3 \\ 0 & -1 & 0 & 0 \\ 3 & 0 & 0 & 1 \end{bmatrix}$$

```
A =
    (2,1)          1
    (4,1)          3
    (1,2)          2
    (3,2)         -1
    (1,3)          5
    (1,4)         -2
    (2,4)         -3
    (4,4)          1
```

Storing a sparse matrix (cont.)

- Compressed sparse column/row (CSC/CSR) form
- Three vectors in CSC
  - `row_ind`: row indices
  - `values`: entries
  - `col_ptr`: pointer to the first element of each column

$$A = \begin{bmatrix} 0 & 2 & 5 & -2 \\ 1 & 0 & 0 & -3 \\ 0 & -1 & 0 & 0 \\ 3 & 0 & 0 & 1 \end{bmatrix}$$

```
row_ind = [2 4 1 3 1 1 2 4]
values = [1 3 2 -1 5 -2 -3 1]
col_ptr = [1 3 5 6 9]
```

MATLAB's sparse technology

- Sparse matrix functions
  - `sparse`, `sprand`
  - `nnz`, `nonzeros`
  - `spy`, `full`, `find`
  - ...
- Run examples in MATLAB's documentation of Sparse Matrices
- J.R. Gilbert, C. Moler, and R. Schreiber, *Sparse Matrices in MATLAB: Design and Implementation.* SIAM Journal on Matrix Analysis, 1992.

Sparsity pattern, structural regularity

- Sparsity pattern of $B = (b_{ij})$ is $A = (a_{ij})$, where

$$a_{ij} = \begin{cases} 1 & \text{if } b_{ij} \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

- Write $A = S(B)$
- A is adjacency matrix where graph theory applies
- B is structurally nonsingular

  $\Leftrightarrow$ W exists s.t. W is nonsingular and $S(W) = S(B)$

  $\Leftrightarrow$ There exists 1-to-1 correspondence between rows and cols

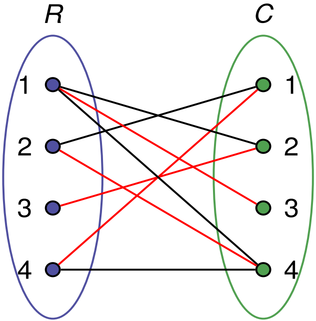
  $\Leftrightarrow$ A transversal T of A exists

  $\Leftrightarrow$ Bipartite graph $\mathcal{B}(A)$ has perfect matching
- Otherwise every W with $S(W) = S(B)$ is structurally singular

Bipartite graph, perfect matching

- A transversal is $T = \{(4,1),(3,2),(1,3),(2,4)\}$
- T corresponds to a perfect matching in $\mathcal{B}(A)$

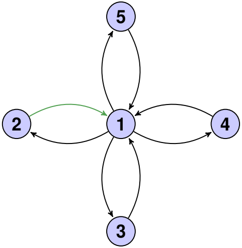
$$A = \begin{bmatrix} 0 & 2 & 5 & -2 \\ 1 & 0 & 0 & -3 \\ 0 & -1 & 0 & 0 \\ 3 & 0 & 0 & 1 \end{bmatrix}$$



Permutations and reordering strategies

- Put a transversal of A on diagonal
- Obtain direcited graph of A
- (i, j) is a directed edge, or an arc, iff. $a_{ij} \neq 0$
- Zero out $a_{21} \Leftrightarrow$ remove arc $(2, 1)$ in $G(V, E)$
- Gaussian elimination removes every arc (i, j) with $i > j$

$$A = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & & & \\ \times & & \times & & \\ \times & & & \times & \\ \times & & & & \times \end{bmatrix}$$

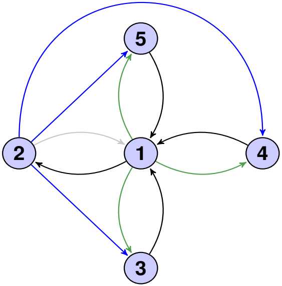


- Removing arc $(2,1)$ creates new arcs in set

$$\{\,(2,j):\ (1,j)\in G(V,E_0),\ j\neq 2\,\} = \{\,(2,3),(2,4),(2,5)\,\}$$

- They correspond to fill-ins in A

$$A=\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & & \times & & \\ \times & & & \times & \\ \times & & & & \times \end{bmatrix}$$



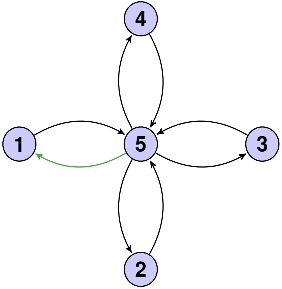
- Properly permuting A gives B

```
B = A([5 2 3 4 1], [5 2 3 4 1])
```

- Zeroing out a_{51}, i.e., removing arc $(5, 1)$, creates NO fill-in:

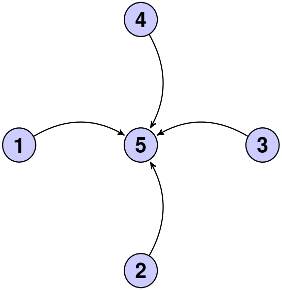
$$\{\,(5, j) : (1, j) \in G(V, E_0),\ j \neq 5\,\} = \emptyset$$

$$B = \begin{bmatrix} \times & & & & \times \\ & \times & & & \times \\ & & \times & & \times \\ & & & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}$$



- Remove arcs $(5,2),(5,3),(5,4)$
- After Gaussian elimination,

$$B' = \begin{bmatrix} \times & & & & \times \\ & \times & & & \times \\ & & \times & & \times \\ & & & \times & \times \\ & & & & \times \end{bmatrix}$$



Reordering algorithms

Consider a structurally symmetric A, i.e., $S(A) = S(A)^T$

Two main goals

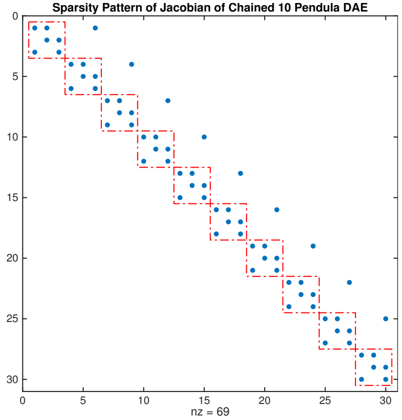
- Reduce bandwidth
  - Reverse Cuthill McKee (RCK) algorithm
  - `symrcm`
- Reduce expected fill-ins
  - Approximate Minimum Degree (AMD) algorithm
  - `symamd`
  - `amd`, `colamd` for asymmetric ones
- Algorithms based on heuristics

Illustration of AMD

```
A = gallery('wathen',50,50);
p = amd(A);
L = chol(A,'lower');
Lp = chol(A(p,p),'lower');

figure;
subplot(2,2,1);    spy(A);
title('Sparsity structure of A');

subplot(2,2,2); spy(A(p,p));
title('Sparsity structure of AMD ordered A');

subplot(2,2,3); spy(L);
title('Sparsity structure of Cholesky factor of A');

subplot(2,2,4); spy(Lp);
title('Sparsity structure of Cholesky factor of AMD ordered A');

set(gcf,'Position',[100 100 800 700]);
```

- Code available in MATLAB's doc of `amd`

Dulmage-Mendelsohn decomposition



- Find permutation matrices P, Q s.t. PAQ is in block triangular form
- Backward substitution by blocks
- `[p,q,r,s]=dmperm(A)`
- `p,q`: permutation vectors
- Permuted matrix is `A(p,q)`
- `r,s`: block boundaries
- E.g., solve ten 3×3's instead of one 30×30

Error analysis of solving linear systems

$$f(\mathbf{x}) = \mathbf{b} - A\mathbf{x} = \mathbf{0}$$

- $\mathbf{x}$ is exact solution; $\widehat{\mathbf{x}}$ is computed
- Relative error $e_{\text{rel}} = \frac{||\mathbf{x} - \widehat{\mathbf{x}}||}{||\mathbf{x}||}$
- Residual

$$\widehat{\mathbf{r}} = f(\widehat{\mathbf{x}}) = \mathbf{b} - A\widehat{\mathbf{x}} = A\mathbf{x} - A\widehat{\mathbf{x}} = A(\mathbf{x} - \widehat{\mathbf{x}})$$

- e_{rel} can be large when $\widehat{\mathbf{r}}$ is small

Condition number

- Relative error is bounded

$$e_{\text{rel}} \leq \kappa_p(A) \cdot \frac{||\widehat{\mathbf{r}}||}{||\mathbf{b}||}$$

where

$$\kappa_p(A) = ||A||_p \cdot ||A^{-1}||_p$$

refers to the condition number (corresponding to some norm p)

- `cond(A)` in MATLAB
- $\kappa(A) \geq 1$; "$=$" holds only if $A = cI_n$ with $c \neq 0$
- Cannot judge $\kappa(A)$ by $\det(A)$

Computing condition number

$$\kappa(A) = ||A|| \cdot ||A^{-1}||$$

- $||A||$ is easy to compute, say in 1-norm or $\infty-$norm
- Need some heuristic to find $||A^{-1}||$
- $Ay = c$, so

  $$||A^{-1}|| \geq \frac{||y||}{||c||},$$

  where c is a vector of ± 1's, with signs chosen to make $||y||$ as large as possible
- Inverse method for finding the smallest singular value of A

Computing condition number (cont.)

Assume a nonsingular A

- $A^T A$ is symmetric positive definite (SPD) with eigenvalues

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n > 0 \tag{1}$$

- $||A||_2 = \sqrt{\lambda_1}$ and $||A^{-1}||_2 = 1/\sqrt{\lambda_n}$ give

$$\kappa_2(A) = \sqrt{\lambda_1/\lambda_n}$$

Assume A is SPD itself, with eigenvalues as (1)

- $||A||_2 = \lambda_1$ and $||A^{-1}||_2 = 1/\lambda_n$ give $\kappa_2(A) = \lambda_1/\lambda_n$
- λ_1 is spectral radius of A, denoted $\rho(A)$