NUMERICKÉ METÓDY LINEÁRNEJ ALGEBRY

07. Riedke matice.

Ing. Marek Macák, PhD.

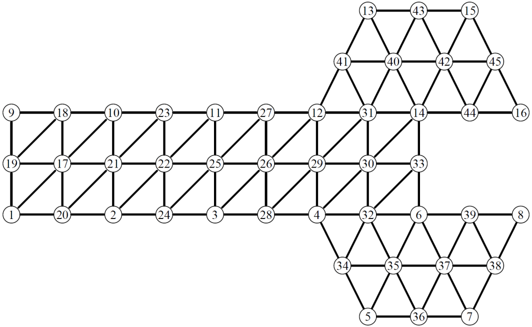
Konzultácie: podľa potreby/dohody

25. Marec 2024

Úvod

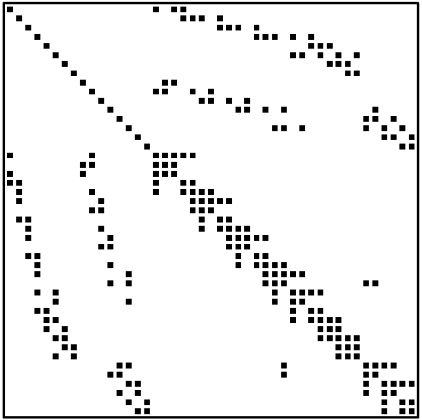
- Vychádzajú z riešenia parciálnych diferenciálnych rovníc.
- V najjednoduchšom prípade ide o diagonálne matrice.
- Hlavný problém riešenia s riedkou maticou, bolo navrhnúť metódy priameho riešenia pre lineárne systémy. Tie museli byť úsporné, z hľadiska pamäťových aj výpočtovej nárokov.
- Pri použití riedkych matíc je možné riešiť väčšie úlohy ktoré by sa pri plnej matici nezmestili do pamäte.

Úvod



Diskretizácia použitím MKP.

Úvod



Riedka matica zo siete MKP.

Úvod

$$A = \begin{pmatrix} a_{11} & 0 & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & 0 \\ 0 & a_{42} & 0 & a_{44} \end{pmatrix}$$

- Pre riedku maticu A sú povolené rovnaké permutácie ako pri plnej matici.
- Ak π je vektor permutácii, potom

$$A_{\pi,\pi} = P_\pi^T A P_\pi \tag{1}$$

kde $P_\pi = I_{\pi,\star}$ a $P_\pi^T = I_{\star,\pi}$.

Úvod

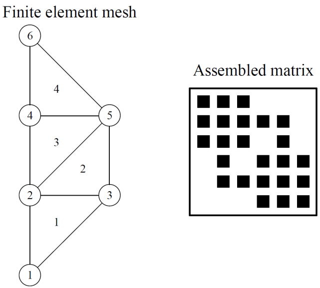
Riadkovo permutovany linearny system pre $\pi = \{1, 3, 2, 4\}$

$$\begin{pmatrix} a_{11} & a_{13} & 0 & 0 \\ 0 & a_{23} & a_{22} & a_{24} \\ a_{31} & a_{33} & a_{32} & 0 \\ 0 & 0 & a_{42} & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_3 \\ x_2 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

Nasledne stlpcovo permutovany linearny system pre $\pi = \{1, 3, 2, 4\}$

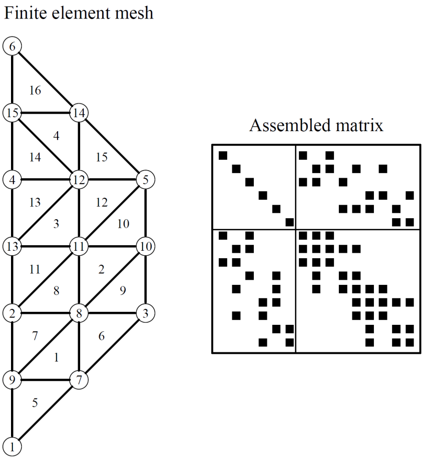
$$\begin{pmatrix} a_{11} & a_{13} & 0 & 0 \\ a_{31} & a_{33} & a_{32} & 0 \\ 0 & a_{23} & a_{22} & a_{24} \\ 0 & 0 & a_{42} & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_3 \\ x_2 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_3 \\ b_2 \\ b_4 \end{pmatrix}$$

Úvod - ako nájsť P_π



(a) Pôvodná sieť

$\Rightarrow$



(b) Zjemnená sieť

Úvod - ako nájsť P_π

- V numerickej lineárnej algebre je Cuthillov-McKeeho algoritmus (CM), pomenovaný podľa Elizabeth Cuthillovej a Jamesa McKeeho, určený na permutáciu riedkej matice, ktorá má symetrický vzor riedkosti, do tvaru pásovej matice s malou šírkou pásu.
- Reverzný Cuthill-McKeeho algoritmus (RCM) predstavil Alan Georgeo a Joseph Liu. Je ten istý algoritmus, ale s obrátenými výslednými indexov.
- Algoritmus Cuthill McKee je variantom štandardného algoritmu prehľadávania podľa šírky, ktorý sa používa v grafových algoritmoch.

Úvod - ako nájsť P_π

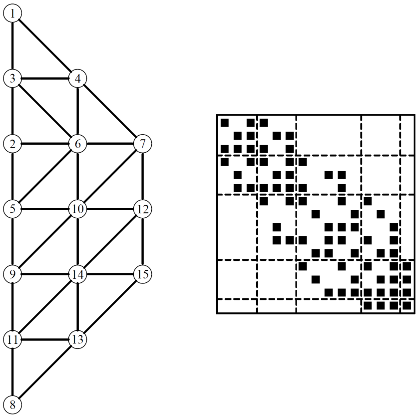
ALGORITHM 3.1: Cuthill-McKee Ordering

Input: initial node i_1*; Output: permutation array* `iperm` .
Start: Set $levset := \{i_1\}$*;* $next = 2$*;*
  Set $marker(i_1) = 1$*;* `iperm` $(1) = i_1$
While $(next < n)$ *Do:*
    $Next\_levset = \emptyset$
    Traverse $levset$ *in order of increasing degree and*
    for each visited node Do:
      For each neighbor i *of* j *such that* $marker(i) = 0$ *Do:*
        Add i *to the set* $Next\_levset$
        $marker(i) := 1$*;* `iperm` $(next) = i$
        $next = next + 1$
      EndDo
    EndDo
    $levset := Next\_levset$
EndWhile

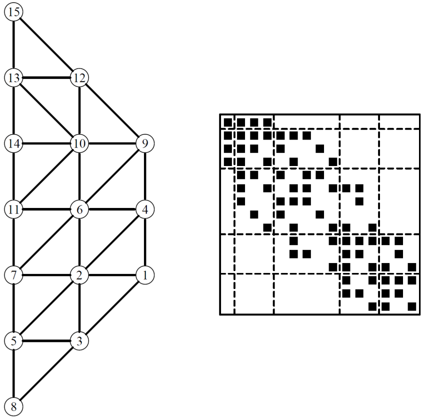
Úvod - ako nájsť P_π

Inými slovami, očíslujte vrcholy podľa určitej štruktúry (napr. vypočítanej prehľadávaním podľa šírky), kde sú vrcholy v každej úrovni navštevované v poradí číslovania ich predchodcov od najnižšieho po najvyššie. Ak sú predchodcovia rovnakí, vrcholy sa rozlišujú podľa stupňa (opäť zoradeného od najnižšieho po najvyšší).

Úvod - ako nájsť P_π

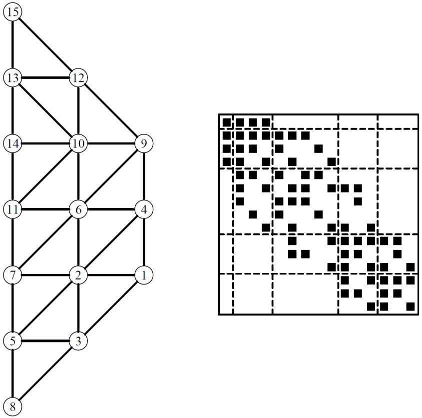


(a) Cuthill-McKee

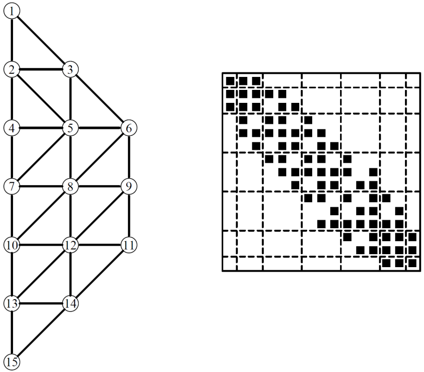


(b) Reverzný Cuthill-McKee

Úvod - ako nájsť P_π



(a) Reverzný Cuthill-McKee



(b) Reverzný Cuthill-McKee, začaty s uzlom 1

Prehľad spôsobov ukladania riedkych matic

- Ak je matica koeficientov A riedka, nulové prvky A nie sú uložené.
- Schémy úložiska prideľujú súvisle miesto v pamäti pre nenulové prvky matice (a možno aj obmedzený počet núl).
- Vyžaduje sa schéma, aby sme vedeli, kde prvky zapadajú do celej matice.
- Existuje mnoho metód na ukladanie riedkych matíc. Budeme hovoriť o komprimovanom ukladaní riadkov a stĺpcov, blokovom komprimovanom ukladaní riadkov, diagonálnom ukladaní a zubatom diagonálnom ukladaní.
  Originál: Compressed Row and Column Storage, Block Compressed Row Storage, Diagonal Storage a Jagged Diagonal Storage.

Compressed Row Storage (CRS)

- Nevytvára absolútne žiadne predpoklady o riedkej štruktúre matice a neuklada žiadne nepotrebné prvky (Najvšeobecnejší formát ukladania).
- Na druhej strane nie sú veľmi efektívne, pretože potrebujú nepriame adresovanie pre každú jednu skalárnu operáciu pri riešení Ax.
- Za predpokladu, že máme nesymetrickú riedku maticu A, vytvoríme 3 vektory: (val), (col_ind) a (row_ind).
  - Vektor (val) uchováva hodnoty nenulových prvkov matice A,
  - Vektor (col_ind) uchováva stĺpcové indexy prvkov vo vektore (val). To znamená, že ak $val(k) = a_{i,j}$, potom $col\_ind(k) = j$.
  - Vektor (row_ptr) uchováva miesta vo vektore val, ktoré začínajú riadok, t. j. ak $val(k) = a_{i,j}$, potom $row\_ptr(i) \leq k < row\_ptr(i+1)$. Podľa konvencie definujeme $row\_ptr(n+1) = nnz + 1$ kde nnz je počet nenulovych prvkov v matici A.
- Úspora pamäte pri tomto prístupe je významná. Namiesto uloženia n^2 prvkov potrebujeme len $2nnz + n + 1$ pamäťových miest.

Compressed Row Storage (CRS)

$$A = \begin{pmatrix} 10 & 0 & 0 & 0 & -2 & 0 \\ 3 & 9 & 0 & 0 & 0 & 3 \\ 0 & 7 & 8 & 7 & 0 & 0 \\ 3 & 0 & 8 & 7 & 5 & 0 \\ 0 & 8 & 0 & 9 & 9 & 13 \\ 0 & 4 & 0 & 0 & 2 & -1 \end{pmatrix}$$

Compressed Row Storage (CRS)

- Formát CRS pre túto maticu je potom určený poliami (*val*), (*col_ind*) a (*row_ind*)

val	10	-2	3	9	3	7	8	7	$3 \cdots 9$	13	4	2	-1
col_ind	1	5	1	2	6	2	3	4	$1 \cdots 5$	6	2	5	6

row_ptr	1	3	6	9	13	17	20

.

Compressed Column Storage (CCS)

- Formát CCS je identický s formátom CRS s tým rozdielom, že namiesto riadkov sa ukladajú (prechádzajú) stĺpce A. Inými slovami formát CCS je formát CRS pre A^T .
- Formát CCS je rovnako špecifikovaný tromi poliami (val), (col_ind) a (row_ind)
- Úspora pamäte je rovnaká ako pri CRS. Namiesto uloženia n^2 prvkov potrebujeme len $2nnz + n + 1$ pamäťových miest (nnz je počet nenulových prvkov v matici A).

Compressed Row Storage (CCS)

$$A = \begin{pmatrix} 10 & 0 & 0 & 0 & -2 & 0 \\ 3 & 9 & 0 & 0 & 0 & 3 \\ 0 & 7 & 8 & 7 & 0 & 0 \\ 3 & 0 & 8 & 7 & 5 & 0 \\ 0 & 8 & 0 & 9 & 9 & 13 \\ 0 & 4 & 0 & 0 & 2 & -1 \end{pmatrix}$$

Compressed Column Storage (CCS)

- Formát CCS pre túto maticu je potom určený poliami (*val*), (*col_ind*) a (*row_ind*)

`val`	10	3	3	9	7	8	4	8	$8 \cdots 9$	2	3	13	-1
`row_ind`	1	2	4	2	3	5	6	3	$4 \cdots 5$	6	2	5	6

`col_ptr`	1	4	8	10	13	17	20

.

Compressed Diagonal Storage (CDS)

- Ak je matica A pásmová pomerne konštantná od riadku k riadku, potom sa oplatí využiť túto štruktúru.
- Matica $A = (a_{i,j})$ je pásmová, ak existujú nezáporné konštanty p, q také, že $a_{i,j} \neq 0$ len vtedy, ak $i - p \leq j \leq i + q$.
- Eliminujeme vektor identifikujúci stĺpec a riadok.
- Schéma ukladania je obzvlášť vhodná, ak matica vzniká z MKP, MKO, MKD.
- V tomto prípade pre maticu A ukladáme pole $val(1:n, -p:q)$.
- CDS zvyčajne zahŕňajú uloženie niekoľkých núl.

Compressed Diagonal Storage (CDS)

$$A = \begin{pmatrix} 10 & -3 & 0 & 0 & 0 & 0 \\ 3 & 9 & 6 & 0 & 0 & 0 \\ 0 & 7 & 8 & 7 & 0 & 0 \\ 0 & 0 & 8 & 7 & 5 & 0 \\ 0 & 0 & 0 & 9 & 9 & 13 \\ 0 & 0 & 0 & 0 & 2 & -1 \end{pmatrix}$$

- Pomocou formátu CDS uložíme túto maticu A do poľa s rozmerom $(6, -1 : 1)$ pomocou mapovania $val(i, j) = a_{i,i+j}$.

Compressed Diagonal Storage (CDS)

- Formát CDS pre túto maticu je potom určený poliami (*val*)

`val(:,-1)`	0	3	7	8	9	2
`val(:, 0)`	10	9	8	7	9	-1
`val(:,+1)`	-3	6	7	5	13	0

Compressed Diagonal Storage (CDS)

$$A = \begin{pmatrix} 10 & -3 & 0 & 1 & 0 & 0 \\ 0 & 9 & 6 & 0 & -2 & 0 \\ 3 & 0 & 8 & 7 & 0 & 0 \\ 0 & 6 & 0 & 7 & 5 & 4 \\ 0 & 0 & 0 & 0 & 9 & 13 \\ 0 & 0 & 0 & 0 & 5 & -1 \end{pmatrix}$$

`val(:,-1)`	0	0	3	6	0	5
`val(:, 0)`	10	9	8	7	9	-1
`val(:,+1)`	0	-3	6	7	5	13
`val(:,+2)`	0	1	-2	0	4	0

Jagged Diagonal Storage (JDS)

- Formát Jagged Diagonal Storage môže byť užitočný pri implementácii iteračných metód na paralelných počítačoch.
- Podobne ako formát CDS ukladá vektory dĺžky veľkosti matice.
- Je pri paralelnej implementácii úspornejší ako CDS z dôvodu menšej komunikacii medzi procesormi.

Jagged Diagonal Storage (JDS)

$$\begin{pmatrix} 10 & -3 & 0 & 1 & 0 & 0 \\ 0 & 9 & 6 & 0 & -2 & 0 \\ 3 & 0 & 8 & 7 & 0 & 0 \\ 0 & 6 & 0 & 7 & 5 & 4 \\ 0 & 0 & 0 & 0 & 9 & 13 \\ 0 & 0 & 0 & 0 & 5 & -1 \end{pmatrix} \longrightarrow \begin{pmatrix} 10 & -3 & 1 & \\ 9 & 6 & -2 & \\ 3 & 8 & 7 & \\ 6 & 7 & 5 & 4 \\ 9 & 13 & & \\ 5 & -1 & & \end{pmatrix}$$

Jagged Diagonal Storage (JDS)

- Formát JDS pre túto maticu je potom určený poliami (*val*) a (*col_ind*).

val(:, 1)	10	9	3	6	9	5
val(:, 2)	−3	6	8	7	13	−1
val(:, 3)	1	−2	7	5	0	0
val(:, 4)	0	0	0	4	0	0

col_ind(:, 1)	1	2	1	2	5	5
col_ind(:, 2)	2	3	3	4	6	6
col_ind(:, 3)	4	5	4	5	0	0
col_ind(:, 4)	0	0	0	6	0	0

Maticovo vektorové násobenie $y = Ax$ alebo $y = A^T x$

- V mnohých iteračných metódach, sa počíta súčin matice aj jej transpozície a vektora.
- To znamená, že vzhľadom na vstupný vektor x chceme vypočítať súčin

$$y = Ax, \quad alebo \quad y = A^T x \tag{2}$$

- Uvedieme algoritmy pre dva z formátov ukladania (CRS a CDS).

CRS - maticovo vektorové násobenie

- Maticovo vektorový súčin $y = Ax$ s použitím formátu CRS možno vyjadriť obvyklým spôsobom:

$$y_i = \sum_j a_{i,j} x_j, \tag{3}$$

  pretože tento postup prechádza riadkami matice A.

- Pre maticu $A^{n \times n}$ je násobenie $y = Ax$ dané:

```
for i = 1, n
   y(i) = 0
   for j = row_ptr(i), row_ptr(i+1) - 1
      y(i) = y(i) + val(j) * x(col_ind(j))
   end
end
```

CRS - maticovo vektorové násobenie

- Metóda násobí iba nenulové položky matice, počet operácií je 2-násobok počtu nenulových prvkov v A, čo je výrazná úspora oproti požiadavke na plnú maticu $2n^2$.
- Pre $y = A^T x$ nemôžeme použiť rovnicu

$$y_i = \sum_j (A^T)_{i,j} x_j = \sum_j a_{j,i} x_j, \tag{4}$$

pretože to znamená prechádzanie stĺpcov matice, čo je pre riadkovo uloženú maticu mimoriadne neefektívna operácia.

CRS - maticovo vektorové násobenie

- Vymenime indexy:

$$pre\,\forall\, j, spocitaj\; pre\,\forall\, i: \qquad y_i \leftarrow y_i + a_{j,i}x_j, \tag{5}$$

- Pre maticu $A^{n\times n}$ je násobenie $y = A^T x$ dané:

```
for i = 1, n
  y(i) = 0
end
for j = 1, n
  for i = row_ptr(j), row_ptr(j + 1) − 1
    y(col_ind(i)) = y(col_ind(i)) + val(i) ∗ x(j)
  end
end
```

CDS - maticovo vektorové násobenie

- Ak je matica $A^{n \times n}$ uložená vo formáte CDS, stále je možné vykonať maticovo-vektorový súčin $y = Ax$ podľa riadkov alebo stĺpcov, ale nevyužije sa tým výhoda formátu CDS.
- Myšlienka je vykonať indexov vo vnorenom cykle nahradením $j \to i + j$ dostaneme:

$$y_i \leftarrow y_i + a_{i,j}x_j \Rightarrow y_i \leftarrow y_i + a_{i,i+j}x_{i+j}. \tag{6}$$

- Algoritmus prechádza cez diagonály $diag = -p, q$, pričom p a q sú (nezáporné) čísla diagonál naľavo a napravo od hlavnej diagonály.
- Hranice pre vnútorný cyklus vyplývajú z požiadavky, že $1 \leq i, i + j \leq n$.

CDS - maticovo vektorové násobenie

- Algoritmus vieme zapísať:

$$
\begin{array}{l}
\textbf{for } i = 1, n \\
\quad y(i) = 0 \\
\textbf{end} \\
\textbf{for } diag = -diag\_left, diag\_right \\
\quad \textbf{for } loc = max(1, 1 - diag), min(n, n - diag) \\
\quad\quad y(loc) = y(loc) + val(loc, diag) * x(loc + diag) \\
\quad \textbf{end} \\
\textbf{end}
\end{array}
$$

CDS - maticovo vektorové násobenie

- $y = A^T x$ je menšou obmenou vyššie uvedeného algoritmu pomocou :

$$\begin{aligned} y_i &\leftarrow y_i + a_{i+1,j} x_j && (7) \\ &= y_i \leftarrow y_i + a_{i+1,i-j} x_{i+j}. && (8) \end{aligned}$$

- Algoritmus vieme potom zapísať:

for $i = 1, n$
  $y(i) = 0$
end
for $diag = -diag\_left, diag\_right$
  for $loc = max(1, 1 - diag), min(n, n - diag)$
    $y(loc) = y(loc) + val(loc + diag, -diag) * x(loc + diag)$
  end
end