F13: Lösning av glesa system av linjära ekvationer på parallelldatorsystem

Huvudsakligen material från Kumar et. al.
Introduction to Parallel Computing, 1st ed Kap. 11

Bo Kågström et al (EE, RG, MR)
2009-05-15

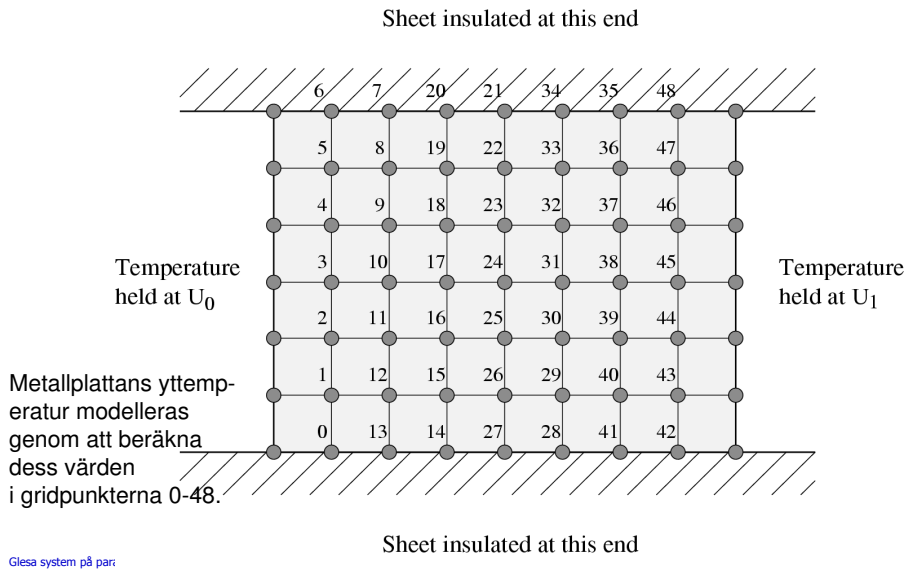
Innehåll

- Glesa matriser och lagringsformat
- Parallella algoritmer för glesa grundläggande operationer
  - Skalärprodukt (sdot)
  - Matrisvektormultiplikation (strukturerad och ostrukturerad gleshet)
- Iterativa metoder för Ax = b – parallella aspekter
  - Jacobi, Gauss-Seidel, konjugerade gradienter (CG), [prekonditionering]
  - Ordning och omordning av rader i glesa A och nätgrafer
- Saker som inte tas upp:
  - Direkta metoder för glesa system (robusta och generella)
    - Omordning (snabbare och mer stabil lösning)
    - Symbolisk faktorisering (sätter upp datastrukturen för matrisfaktoriseringen, allokerar minne, fill-in)
    - Numerisk faktorisering

Parallella algoritmer för glesa lin. ekv-system

- Fysiskt system representerat av en matematisk modell
- Kontinuerlig domän (yta, rum etc) diskretiseras ofta med ett "nät" av punkter → ändligt antal beräkningspunkter
- Samband i den diskretiserade modellen ger ofta endast en liten andel nollskilda element i matriserna
- Beräkningar mycket effektivare och minnessnålare om glesheten utnyttjas
- Ofta skulle inte lagringsutrymmet räcka till för att lagra matriserna på vanligt sätt (som täta matriser)

Diskretiserad domän



Lagring av glesa matriser: koordinatformat

q = # nollskilda el. << n^2

Nollskilda element – godt. ordning

$$\begin{pmatrix} 1 & 0 & 0 & 2 & 0 & 3 \\ 4 & 5 & 0 & 0 & 0 & 0 \\ 0 & 6 & 7 & 0 & 0 & 8 \\ 9 & 0 & 0 & 10 & 11 & 12 \\ 0 & 13 & 0 & 0 & 14 & 0 \\ 0 & 0 & 0 & 0 & 0 & 15 \end{pmatrix}$$

(a) A sparse matrix nxn

	VAL														
qx1	8	6	12	1	15	14	9	2	3	5	13	4	11	7	10
	I														
	2	2	3	0	5	4	3	0	0	1	4	1	3	2	3
	J														
	5	1	5	0	5	4	0	3	5	1	1	0	4	2	3

Radkoordinater

Kolumnkoordinater

(b) Storage in coordinate format

Figure 11.2 A 6×6 sparse matrix and its representation in the coordinate storage format.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

Komprimerad radlagring

VAL	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
J	0	3	5	0	1	1	2	5	0	3	4	5	1	4	5
I	0	3	5	8	12	14									

Nollskilda element i radordning

Motsvarande kolumnindex

Pekare till i:te radens 1:a el. i VAL och J

ure 11.3 CSR storage of the 6×6 sparse matrix of Figure 11.2(a).
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

- Även komprimerad kolumnlagring (CSC) – rader och kolumner byter roll i lagringen (också kallat *Harwell-Boeing*-format)

Diagonallagringsformat

d = #diagonaler = 4

$$\begin{pmatrix} 1 & 2 & 0 & 0 & 0 & 0 \\ 3 & 4 & 5 & 0 & 0 & 0 \\ 0 & 6 & 7 & 0 & 0 & 0 \\ 8 & 0 & 9 & 10 & 11 & 0 \\ 0 & 13 & 0 & 0 & 14 & 15 \\ 0 & 0 & 16 & 0 & 17 & 18 \end{pmatrix}$$

(a) A sparse matrix

Nollskilda element diagonalvis i varje kolumn

Avstånd till huvud-diagonalen

nxd

VAL				
-	-	1	2	
-	3	4	5	
-	6	7	0	
8	9	10	11	
13	0	14	15	
16	17	18	-	

dx1	OFFSET			
	-3	-1	0	1

(b) Storage in diagonal format

Figure 11.4 A sparse matrix stored in the diagonal format.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

Ellpack-Itpack-format (E-I-format)

Bra när m = max(#nz) i ngn rad inte mkt större än mean(#nz) i ngn rad

$$\begin{pmatrix} 1 & 2 & 0 & 0 & 0 & 0 \\ 3 & 4 & 5 & 0 & 0 & 0 \\ 0 & 6 & 7 & 0 & 0 & 0 \\ 8 & 0 & 9 & 10 & 11 & 0 \\ 0 & 13 & 0 & 0 & 14 & 15 \\ 0 & 0 & 16 & 0 & 17 & 18 \end{pmatrix}$$

(a) A sparse matrix

Nollskilda element radvis

Motsvarande kolumnindex

nxm

VAL				J
1	2	-	-	0 1 -1 -
3	4	5	-	0 1 2 -1
6	7	-	-	1 2 -1 -
8	9	10	11	0 2 3 4
13	14	15	-	1 4 5 -1
16	17	18	-	2 4 5 -1

(b) Storage in Ellpack-Itpack format

-1 signalerar radslut

Figure 11.5 A sparse matrix stored in Ellpack-Itpack format.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

Jagged-diagonal-format

Först: Matrisens rader ordnas efter minskande antal nollskilda element

$$\begin{pmatrix} 1 & 0 & 0 & 2 & 0 & 3 \\ 4 & 5 & 0 & 0 & 0 & 0 \\ 0 & 6 & 7 & 0 & 0 & 8 \\ 9 & 0 & 0 & 10 & 11 & 12 \\ 0 & 13 & 0 & 0 & 14 & 0 \\ 0 & 0 & 0 & 0 & 0 & 15 \end{pmatrix} \Rightarrow \begin{pmatrix} 9 & 0 & 0 & 10 & 11 & 12 \\ 1 & 0 & 0 & 2 & 0 & 3 \\ 0 & 6 & 7 & 0 & 0 & 8 \\ 4 & 5 & 0 & 0 & 0 & 0 \\ 0 & 13 & 0 & 0 & 14 & 0 \\ 0 & 0 & 0 & 0 & 0 & 15 \end{pmatrix}$$

(a) A sparse matrix (b) The matrix with reordered rows

Nollskilda element i (b)-matrisen i "jagged diagonal"-ordning

Kolumnindex

Pekare till startindex för varje diagonal

VAL	9	1	6	4	13	15	10	2	7	5	14	11	3	8	12
J	0	0	1	0	1	5	3	3	2	1	4	4	5	5	5
I	0	6	11	14											

(c) Storage in jagged-diagonal format

qx1 (q = # nz)

Figure 11.6 The jagged-diagonal storage scheme.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

Parallell skalär-produkt

Vanlig i algoritmer för glesa matriser (vektorerna är ej glesa)

- Om x och y av längd n är uniformt fördelade på processorerna gör varje processor n/p *multiplikationer* och n/p-1 *additioner* varpå det krävs en *global summering*
- $T_p = 2n/p + (t_s+t_w)\log p$

```
1.  procedure INNER_PRODUCT (x, y, a, n)
2.  begin
3.      a := 0;
4.      for i := 0 to n − 1 do
5.          a := a + x[i] × y[i];
6.  end INNER_PRODUCT
```

Program 11.1 An algorithm for computing the inner product of two dense $n \times 1$ vectors x and y.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

Matrisvektormultiplikation - matrisstruktur

- y = Ax, A gles nxn, x tät nx1 ➔ y tät
- Kostsammaste operationen i de flesta (iterativa) algoritmer för lösning av linjära ekvationssystem
- Matriserna har ofta olika typer av struktur, t.ex.
  - ett *fåtal diagonaler* nära huvuddiagonalen
  - *ostrukturerade* (närmast slumpvis placering av element)
  - *bandmatriser*: nollskilda element är samlade inom ett band nära huvuddiagonalen (men ostrukturerat inom bandet)
  - symmetriska
- Strukturer vill man utnyttja om det går

Block-tridiagonala matriser

- Antag diskretiserat nät för PDE (Laplace)
- Gridpunkterna numreras radvis från 0 till n-1
- Finit differens-approximation av derivatorna ger:

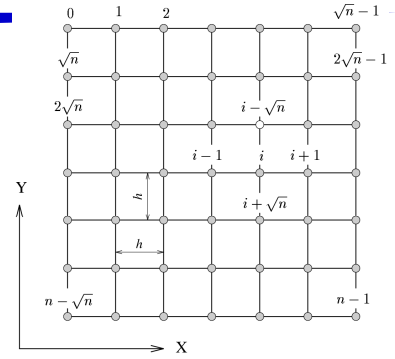


Figure 11.8 A $\sqrt{n} \times \sqrt{n}$ grid with natural ordering of grid points.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

Generellt gäller för rad i:

$$a_i x[i-\sqrt{n}] + b_i x[i-i] + c_i x[i] + d_i x[i+1] + e_i x[i+\sqrt{n}] = f_i$$

Om vi låter koefficienter med index i representera element i rad i i matrisen erhålls maximalt 5 element per rad

Block-tridiagonala matriser - exempel

n = #gridpunkter = 16 blockstorlek = √n x √n

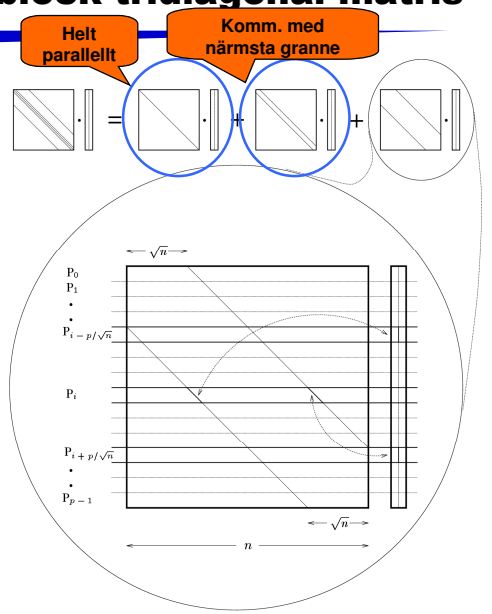
$$\left[\begin{array}{cccc|cccc|cccc|cccc}
c_0 & d_0 & & & e_0 & & & & & & & & & & & \\
b_1 & c_1 & d_1 & & & e_1 & & & & & & & & & & \\
 & b_2 & c_2 & d_2 & & & e_2 & & & & & & & & & \\
 & & b_3 & c_3 & & & & e_3 & & & & & & & & \\
\hline
a_4 & & & & c_4 & d_4 & & & e_4 & & & & & & & \\
 & a_5 & & & b_5 & c_5 & d_5 & & & e_5 & & & & & & \\
 & & a_6 & & & b_6 & c_6 & d_6 & & & e_6 & & & & & \\
 & & & a_7 & & & b_7 & c_7 & & & & e_7 & & & & \\
\hline
 & & & & a_8 & & & & c_8 & d_8 & & & e_8 & & & \\
 & & & & & a_9 & & & b_9 & c_9 & d_9 & & & e_9 & & \\
 & & & & & & a_{10} & & & b_{10} & c_{10} & d_{10} & & & e_{10} & \\
 & & & & & & & a_{11} & & & b_{11} & c_{11} & & & & e_{11} \\
\hline
 & & & & & & & & a_{12} & & & & c_{12} & d_{12} & & \\
 & & & & & & & & & a_{13} & & & b_{13} & c_{13} & d_{13} & \\
 & & & & & & & & & & a_{14} & & & b_{14} & c_{14} & d_{14} \\
 & & & & & & & & & & & a_{15} & & & b_{15} & c_{15}
\end{array}\right]$$

√n block på huvuddiagonalen; √n – 1 block på sub- och superdiagonalerna

Hur göra parallell matris-vektor-multiplikation med denna struktur?

Matris-vektormult. för block-tridiagonal matris

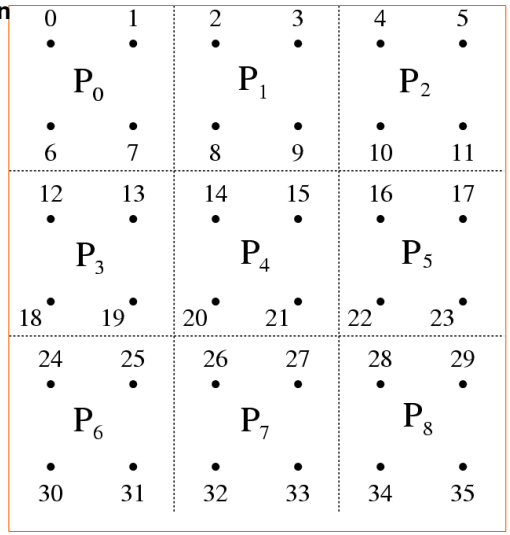
- Antag ”block-striped partitioning” av både matrisen och vektorn och p ≤ √n (annars blir det lite krångligare och *mycket* sämre)
- Varje rad i matrisen kräver 5 vektor-element för sin delberäkning
- Elementen på huvuddiagonalen ligger rätt – trivialt parallellt
- Elementen på sub- och superdiagonalerna ligger på processorgranne – komm. $2(t_s+t_w)$
- Element på avlägsna diagonalerna (+/- √n) måste utbytas också – kommunikation $2(t_s+t_w\sqrt{n})$
- Beräkningar kostar $5t_a*n/p$
- $T_p = 5t_an/p + 2(t_s+t_w\sqrt{n})$
- Isoeffektivitetsfunktionen: $\Theta(p^2)$



Bättre partitionering av block-tridiagonal matris

För p > √n är följande partition av beräkningsnätet avsevärt bättre (n = 36, p = 9):

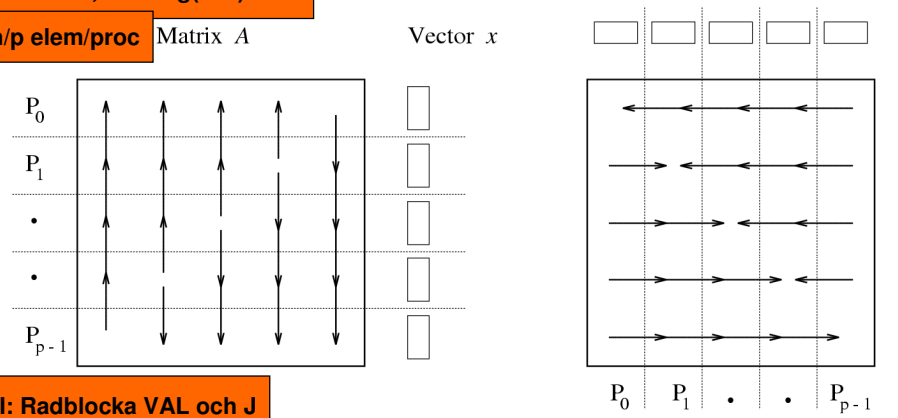
- Matrisrader som hör till punkter inom given partition hamnar på samma proc. – samma med vektorn
- √(n/p) (= 2) block av √(n/p) rader (+ vektor-element)
- Vektorelement motsvarande randpunkter för partitionen utbytes med *logiska grannar*: $4(t_s+t_w\sqrt{(n/p)})$
- $T_p = 5t_an/p + 4(t_s+t_w\sqrt{(n/p)})$
- Iso. eff. funk: $\Theta(p^2)$



Matris-vektormult. för ostrukturerad matris

nxn matris, m = avg(#nz)/rad

mn/p elem/proc



E-I: Radblocka VAL och J

(a) All-to-all broadcast with rowwise striping

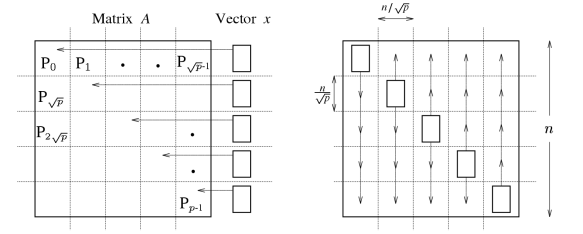
(b) Multinode accumulation with columnwise striping

$T_p = t_amn/p + t_s\log p + t_wn = \Theta(T_s)$ ☹!

Liknande problem!!

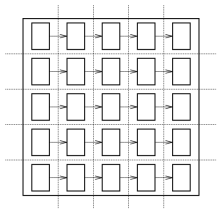
Snabbare algoritm för ostrukturerade matriser

- 2D-blockning
- Vektorn hör till sista proc.-kolumnen
- Alignment-operation av x
- One-to-all broadcast av x i processor-kolumnerna
- Single-node ackumulering av delresultat
- $T_p = t_a mn/p + t_s \log p + (3/2)t_w n \log p/\sqrt{p} \leq \Theta(T_s)$
- Dock inte speciellt skalbar eller kostnadsoptimal!
- För bättre algoritmer krävs viss struktur (t ex symmetri)

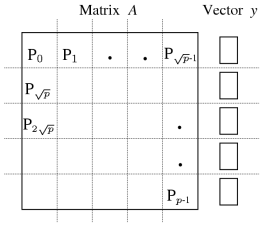


(a) Initial data distribution and communication steps to align the vector along the diagonal

(b) One-to-all broadcast of portions of the vector along processor columns

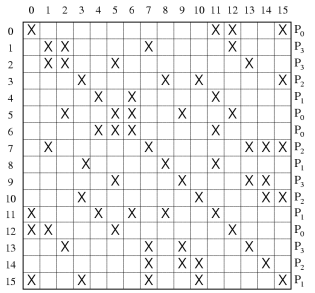


(c) Single-node accumulation of partial results

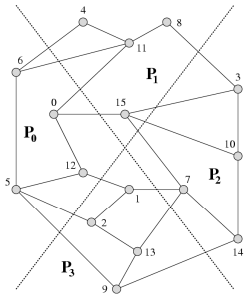


(d) Final distribution of the result vector

Gles matris och dess grafrepresentation



(a) A 16 x 16 symmetric random sparse matrix



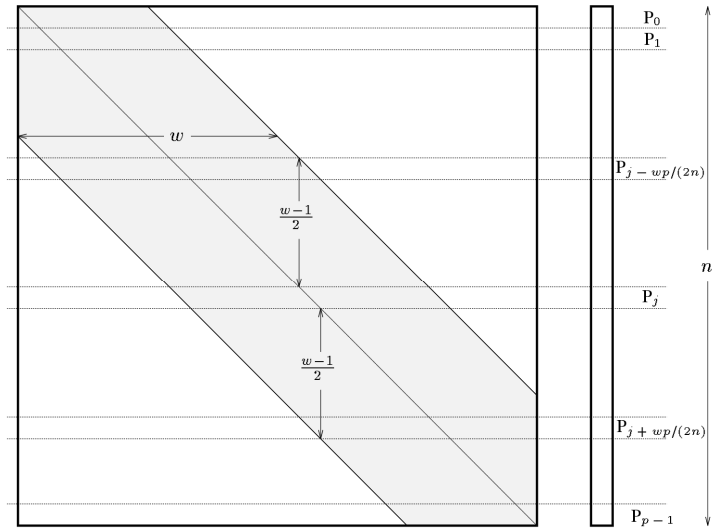
(b) The associated graph and its four partitions

Figure 11.12 A 16×16 unstructured sparse matrix with symmetric structure and its associated graph partitioned among four processors.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

- Beroenden mellan element i matrisen framgår av den graf som ges av en grafs "adjacency"-matris
- För att minimera kommunikation så partitioneras grafen enligt god heuristik
- Inga detaljer här!!!

Matris-vektormult. - ostrukturerad bandmatris

- w – bandbredd
- Ellpack-Itpack
- Utbyte av vektor-element för att klara beräkningarna
- $T_p = t_a mn/p + t_s wp/n + t_w w$
- Kostnadsoptimal för p = O(mn/w)
- Slutsats: sämre skalbarhet för större bandbredd!



Iterativa metoder för Ax = b

- Vid iterativ lösning av linjära ekvationssystem Ax = b genereras en sekvens av approximationer x_k till en lösning x
- I varje iteration används matrisen A i en eller flera *matris-vektormultiplikationer* (gles GEMV)
- Antalet iterationer för att lösa problemet beror på metod, egenskaper hos A och önskad noggrannhet på lösningen
- Andra vanliga operationer: *skalärprodukter* (sdot), saxpy
- Prestandaanalys görs *per iteration*

Jacobi

```
1.   procedure JACOBI_METHOD (A,b,x,ε)
2.   begin
3.       k := 0;
4.       Select initial solution vector x_0;
5.       r_0 := b − Ax_0;
6.       while (||r_k||_2 > ε||r_0||_2) do
7.       begin
8.           k := k + 1;
9.           for i := 0 to n − 1 do
10.              x_k[i] := r_{k−1}[i]/A[i,i] + x_{k−1}[i];   /* Equation 11.17 */
11.          r_k := b − Ax_k;
12.      endwhile;
13.      x := x_k;
14.  end JACOBI_METHOD
```

Betrakta A = D + M, där D är diagonal och M resten. Då är Jacobis metod:

$x_{k+1} = - D^{-1}(Mx_k + b)$

Kräver diagonaldominant A för att konvergera

Program 11.2 The serial Jacobi iterative method for solving a system of linear equations.

Glesa Copyright (r) 1994 Benjamin/Cummings Publishing Co.

Parallell Jacobi

```
1.   procedure JACOBI_METHOD (A,b,x,ε)
2.   begin
3.       k := 0;
4.       Select initial solution vector x_0;
5.       r_0 := b − Ax_0;
6.       while (||r_k||_2 > ε||r_0||_2) do
7.       begin
8.           k := k + 1;
9.           for i := 0 to n − 1 do
10.              x_k[i] := r_{k−1}[i]/A[i,i] + x_{k−1}[i];   /* Equation 11.17 */
11.          r_k := b − Ax_k;
12.      endwhile;
13.      x := x_k;
14.  end JACOBI_METHOD
```

Kräver kommunikation!

Utföres helt parallellt utan kommunikation!!!

Kräver kommunikation!

Program 11.2 The serial Jacobi iterative method for solving a system of linear equations.

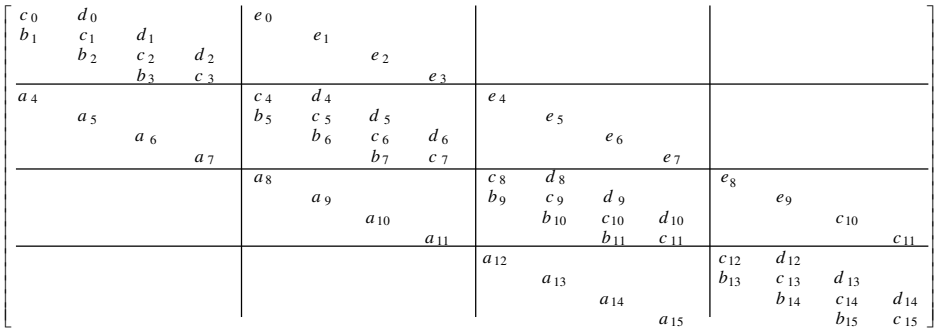
Glesa Copyright (r) 1994 Benjamin/Cummings Publishing Co.

Parallell Gauss-Seidel

- Intuitivt i Jacobi: beräkning av x måste ske sekventiellt eftersom x[i] beror av x[0], x[1] ... ,x[i-1]
- För gles A är det dock ej säkert att det finns beroende till alla förgående x-värden
- Om A[i,j] = 0 saknar x[i] beroende av x[j]
  → x[i] kan beräknas så snart alla x[j] för j < i och A[i,j] ≠ 0 har beräknats → flera x-värden kan beräknas parallellt
- Gauss-Seidel återanvänder nya värden så fort de beräknats och utför två Jacobi-steg i varje iteration:
  - **A = D-L-M**, **D** = diagonal, **L** = strikt nedre triangulär, **M** = resten
  - **$x_{k+1} = (D-L)^{-1}(Mx_k + b)$**

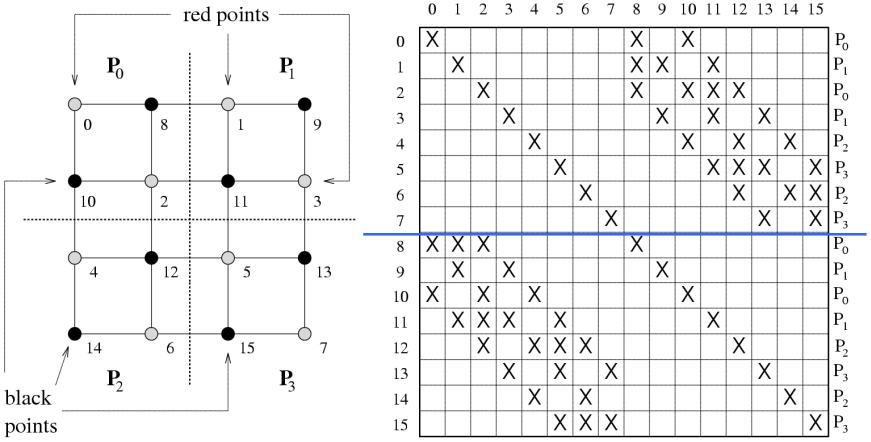
Radernas ordning spelar roll för beroenden

- I Jacobi: beräkning av x[i] beror av x[i-1] och x[i-√n]
- Eftersom x[i-1] beräknas i iterationen före x[i] kan dessa beräkningarna ej parallelliseras (beroende på radernas ordning!)
- Lösning: beräkna oberoende punkter (icke grannar) parallellt !



Röd-svart ordning

Gauss-Seidel utnyttjar implicit röd-svart omordning – först beräknas röda, sedan svarta punkter.



(a) A 4 X 4 grid with red-black ordering (b) Corresponding coefficient matrix

Figure 11.14 A partitioning among four processors of a 4 × 4 finite difference grid with

Flerfärgad ordning för godtyckliga matriser

Generalisering av Gauss-Seidel

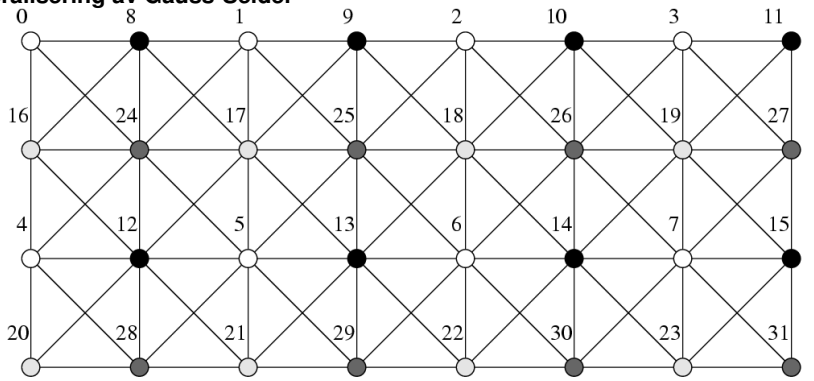


Figure 11.15 Multicolored ordering of a finite element graph using four colors.

Copyright (r) 1994 Benjamin/Cummings Publishing Co.

Konjugerade gradientmetoden (CG-metoden)

- Mest använda metoden för att iterativt lösa $Ax = b$ då
  - A är symmetrisk ($A = A^T$) och
  - positivt definit ($x^TAx > 0$, för alla vektorer $x <> 0$)
- Finner minimum till
  $$q(x) = (1/2)\, x^TAx - x^Tb$$
- Gradienten (derivatan) till $q(x)$ är $Ax - b$ ($=0$ i minimipunkten)
- I iteration k beräknas en sökriktning p_k och en steglängd σ_k som som minimerar q längs p_k
- Ny x-vektor beräknas $x_k = x_{k-1} + \sigma_k p_k$
- Residualen kan uppdateras $r_k = r_{k-1} - \sigma_k A p_k$
- Beräkningarna avslutas då residualen tillräckligt liten!

Parallella konjugerade gradientmetoden

- Ingredienser i PCG:
  - SAXPY (single prec. ax plus y)
  - Skalär-produkter
  - Matris-vektormultiplikation
  - Lösning av linjära system
- Parallellisering sker med metoder vi nämnt (eller inte nämnt)

Finita element-metoden (FEM)

- Beräkna approximativa numeriska lösningar till PDE över diskret värderum
- Till skillnad från i det finita differensnätet ska varje nätpunkt utbyta information med varje punkt som den delar ett element med (totalt 9)
- Styvhetsmatris A beräknas mha ett antal integraler över grafens element (A[i,j] ≠ 0 om nätpunkt i och j har element gemensamt)
  → ekvationssystem Ax = b

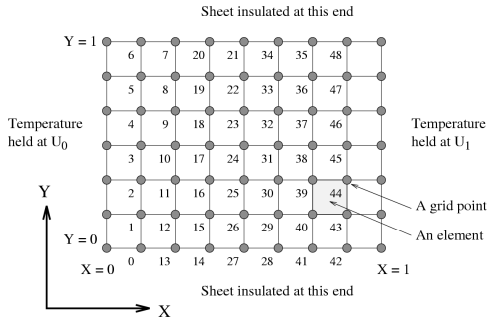


Figure 11.17 A grid dividing a sheet of metal into a finite element mesh of 48 elements. The internal nodes or grid points at which the coefficients must be determined are labeled 0 through 48. The nodes at the periphery are not labeled because the temperature at these points is determined by boundary conditions.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

Styvhetsmatrisen

- I de flesta tillämpningar är grafen mycket irreguljär => ostrukturerad gles matris
- Beräkning av styvhetsmatris och kraftvektor är billigt och kan göras lokalt av processor som äger respektive nätpunkter
- Linjära ekvationssystemet blir stort och glest
  => hur det löses avgör prestanda på hela FEM
- Antag att Ax = b löses med CG:
  - SAXPY ger normalt ingen kommunikation
  - Dot-produkt: O(log p) med CT routing
  - Matris-vektormultiplikation: kommunikation beror av processorns antal nätpunkter som delar element med annan processors nätpunkter
- Prestanda avgörs av hur beräkningsnätet partitioneras!

1-dimensionell partitionering för nätgrafer

Optimal partitionering NP-hard!

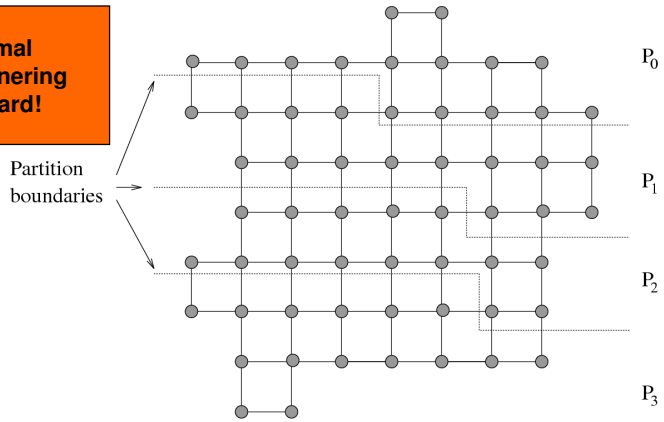


Figure 11.18 One-dimensional striped partitioning of a finite element mesh among four processors.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

2-dimensionell partitionering för nätgrafer

- Vertikal och horisontell partitionering läggs samman
- Ger ej alltid samma antal noder per partition

=> Gör gränsförfining

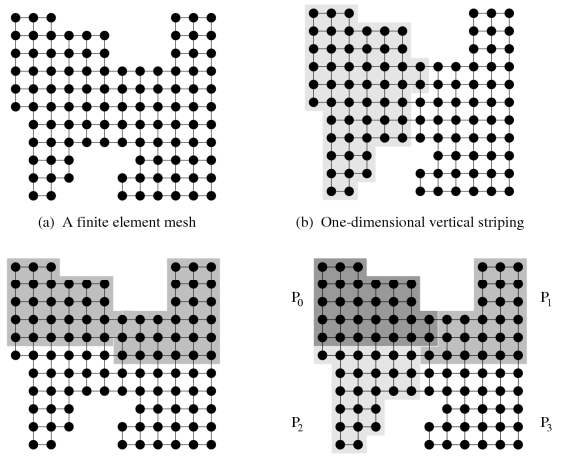


Figure 11.19 Two-dimensional striping of a mesh graph for a 2×2 mesh of processors.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.

Blockpartitionering av godtyckliga grafer

- Grafen delas upp i nivåer
- Processorer tilldelas noder i nivåordning

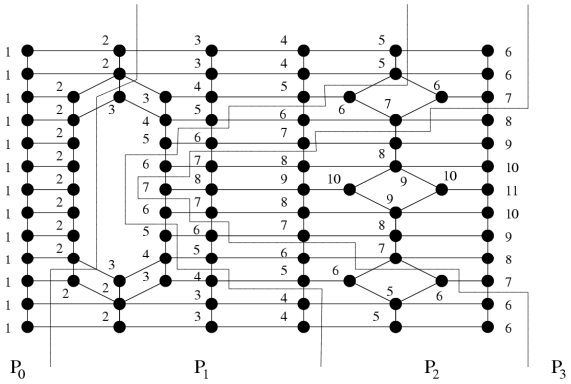


Figure 11.20 The levelization process for a generalized finite element graph. The levels partition the graph into four one-dimensional stripes that are assigned to four processors.
Copyright (r) 1994 Benjamin/Cummings Publishing Co.