PARALLEL ~~SEQUENTIAL~~ NESTED DISSECTION FOR SOLVING SYSTEM $Ax=b$

Outline of the talk:

1. Sequential solving
2. Parallelization
3. CM-implementation:
   - data mapping
   - algorithm
   - implementation of primitives (matrix operations in CM).

Lena Nekludova
TMC
June 1986

This is our problem:

Solve an equation $Ax = b$, A - $n \times n$ matrix, b - n-vect

where (1) A is symmetric positive definite $\Leftrightarrow$

$\Leftrightarrow$ for any subset of variables, corresponding submatrix of A has det > 0. E.g:

$$A = \begin{array}{c} v_1 \\ v_2 \\ v_3 \end{array} \begin{bmatrix} 1 & -5 & 3 \\ -5 & 2 & 0 \\ 3 & 0 & 10 \end{bmatrix} \xrightarrow{\{v_1, v_3\}} \begin{bmatrix} 1 & 3 \\ 3 & 10 \end{bmatrix}; \; det = 1 >$$

$$\xrightarrow{\{v_1\}} [1]; \; det = 1 > 0.$$

(2) A is sparse $\Leftrightarrow$ $O(n)$ elements are nonzeroes.

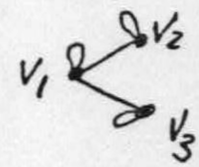
(3) $n > 1000$

(4) Graph(A) is known beforehand.

<u>Def:</u> Incidence graph Graph(A) of matrix A:

Its vertices correspond to variables of A

—"— edges —"— to nonzero elements of A

E.g: for A above, Graph(A) = 

<u>Note</u>: in most examples, I will be using matrices with Graph(A) = grid.

1) SEQUENTIAL SOLVING of $Ax=b$

DIRECT INVERSION: $x = A^{-1}b$.

CLAIM: $\exists!$ DECOMPOSITION $A = L \cdot D \cdot U$,

where $L = \begin{bmatrix} 1 & & 0 \\ & 1 & \\ * & & 1 \end{bmatrix}$, $U = \begin{bmatrix} 1 & & * \\ & 1 & \\ 0 & & 1 \end{bmatrix}$, $D = \text{diag}$.

COR: $A^{-1} = U^{-1}D^{-1}L^{-1} = (L^T)^{-1}D^{-1}L^{-1}$

HOW TO FIND L^{-1} and D^{-1}?

BY GAUSSIAN ELIMINATION.

GAUSSIAN ELIMINATION (NO PIVOTING)

$$\left[\begin{array}{cccc} 1 & & & 0 \\ & 1 & & \\ & & \ddots & \\ 0 & & & 1 \end{array}\right] \Bigg| \left[\begin{array}{cccc} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & \cdots & \cdots & \cdots \\ \vdots & & & \\ a_{n1} & \cdots & \cdots & a_{nn} \end{array}\right] \xrightarrow{1^{st}\ \text{COLUMN}}$$

$$\left[\begin{array}{cccc} 1 & 0 & \cdots & 0 \\ -a_{21}/a_{11} & 1 & 0 \cdots & 0 \\ \vdots & & \ddots & \\ -a_{n1}/a_{11} & & & 1 \end{array}\right] \Bigg| \left[\begin{array}{cccc} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & & & \\ \vdots & & * & \\ 0 & & & \end{array}\right] \xrightarrow{2^{nd}} \text{COLUMN}$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$\underbrace{\left[\begin{array}{cccc} 1 & & & 0 \\ & 1 & & \\ & & \ddots & \\ * & & & 1 \end{array}\right]}_{=L^{-1}} \Bigg| \left[\begin{array}{cccc} a_{11} & & & * \\ & d_{22} & & \\ & & \ddots & \\ 0 & & & d_{nn} \end{array}\right] = D$$

$$A^{-1} = (L^T)^{-1} D^{-1} L^{-1}$$

L^{-1} CAN BE DENSE EVEN FOR SPARSE L.

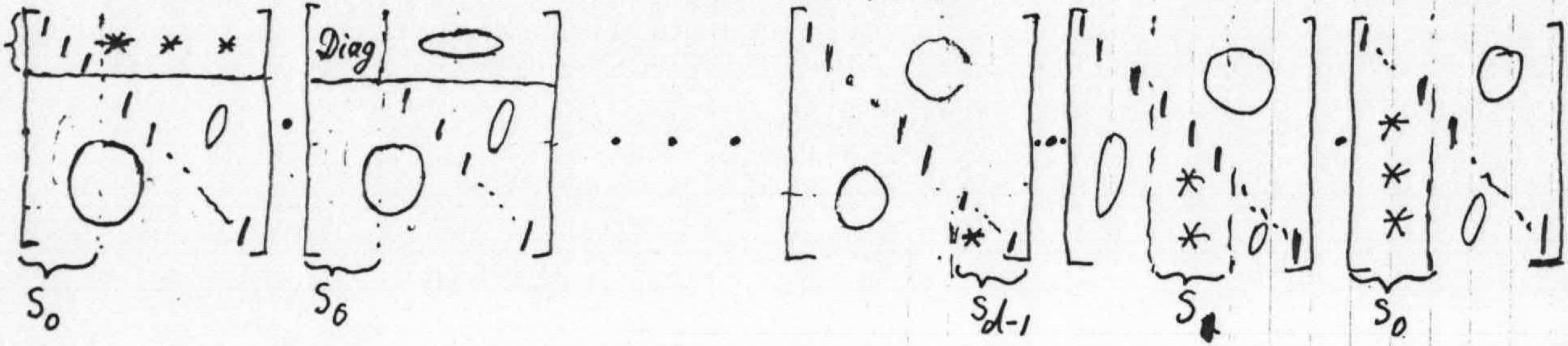
Seq: $O(n^3)$

Par: SPACE/TIME = $O(n^2)/O(n \log n)$.

Generalization of LDU decomposition of A^{-1}

Decompose A^{-1} into $3d$ factors:

$$(L_1^T)^{-1} \cdot \cdots \cdot \mathcal{D}_1^{-1} \cdot \cdots \cdot (L_{d+1}^T)^{-1} \cdot \mathcal{D}_{d+1}^{-1} \cdot (L_{d+1})^{-1} \cdot \cdots \cdot (L_2)^{-1} \cdot (L_1)^{-1}$$

$^{-1} =$ 

Then $x = A^{-1}b$

This decomposition is obtained by partial G.E. on $S_0, S_1, \dots$

Here $S_0, S_1, \dots, S_{d-1}$ are some subgroups of variables of A

We want to choose these subgroups so that all L_i are sparse (i.e., each L_i has only $O(n)$ nonzero variables.

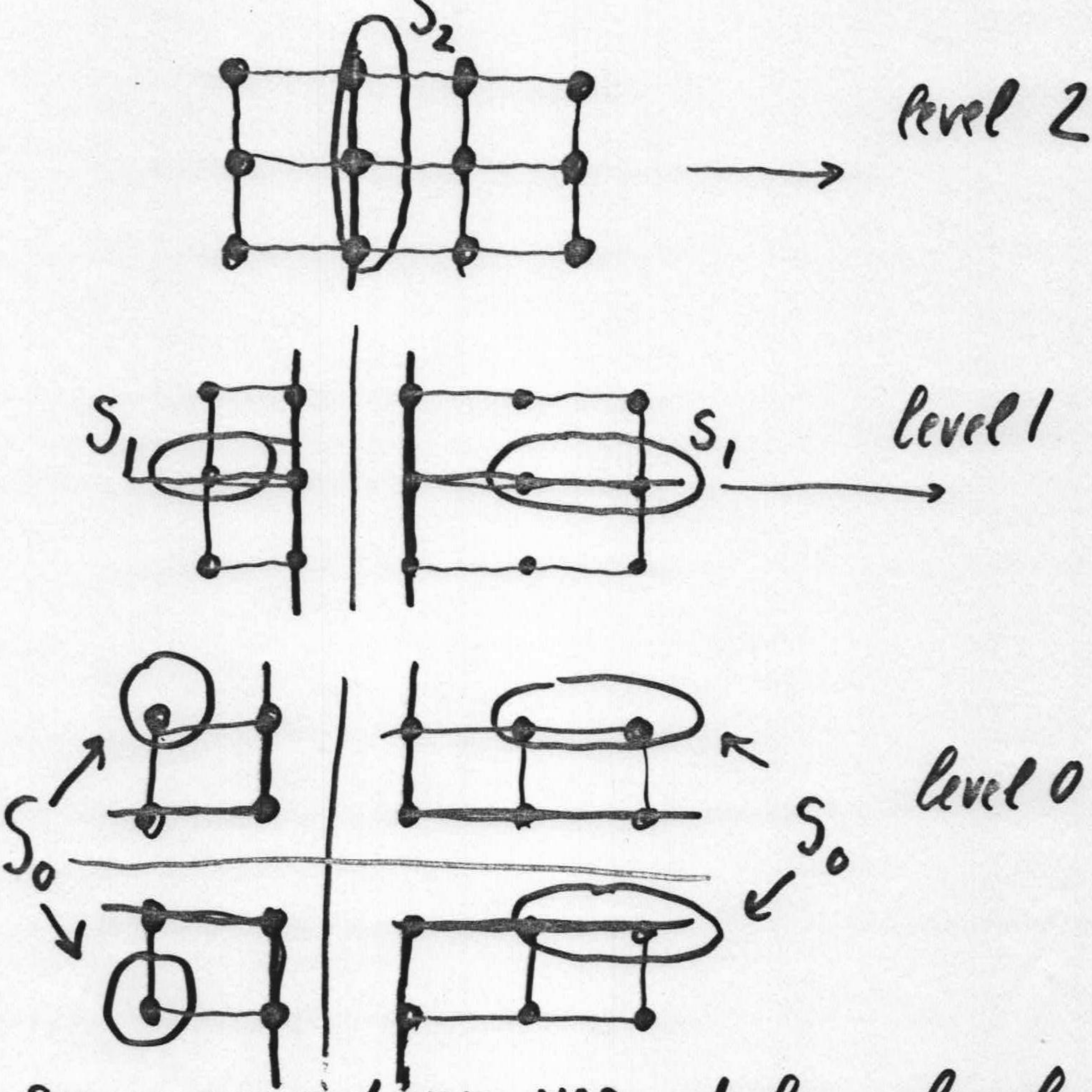
FOR MANY TYPES OF GRAPH(A)
(e.g. for all planar graphs)
we ~~shall~~ can choose $S_0, \ldots, S_d$
in SOME special way.

This will make all matrices in (α)
<u>sparse</u>.

NAMELY, $S_0, \ldots, S_d$ WILL be constructed
with the help of SEPARATOR TREE of A

(NO PRECISE DEFINITION GIVEN HERE).

$O(\sqrt{n})$ - SEPARATOR FOR GRAPH/MATRIX



SEPARATOR TREE HAS $d \sim \log n$ levels

At level i it gives decomposition

of Graph(A) into 2^{d-i-1} subgraphs (not disjoint)

Claim: IF WE CHOOSE i-th SEPARATOR FOR S_i, then each factor IN (α) has $O(n)$ nonzero elements.

MOREOVER, WE CAN FIGURE OUT their POSITIONS BY looking at the sep. tree.

SEQUENTIAL NESTED DISSECTION:

— CONSTRUCT SEPARATOR TREE FOR $G(A)$

— FACTOR A^{-1} INTO PRODUCT OF $\sim 2\log n$ SPARSE $n \times n$ matrICES.

— SOLVE: $x = A^{-1}b$

SEQ: $O(n^{3/2})$

?! PAR: $\text{SPACE}/\text{TIME} \stackrel{?}{=} O(n)/O(\sqrt{n})$

NEED PARALLEL NESTED DISSECTION

That is, WANT to factor A^{-1} in parallel

Recall: factorISATION of A^{-1} is obtained by partial Gaussian elimination

Recursive factorization of A^{-1} by partial G.E.

want: $$A^{-1} = (L_1^T)^{-1} \mathcal{D}_1^{-1} \cdots (L_{d-1}^T)^{-1} \mathcal{D}^{-1} L_{d-1}^{-1} \cdots L_2^{-1} L_1^{-1} \quad (\alpha)$$

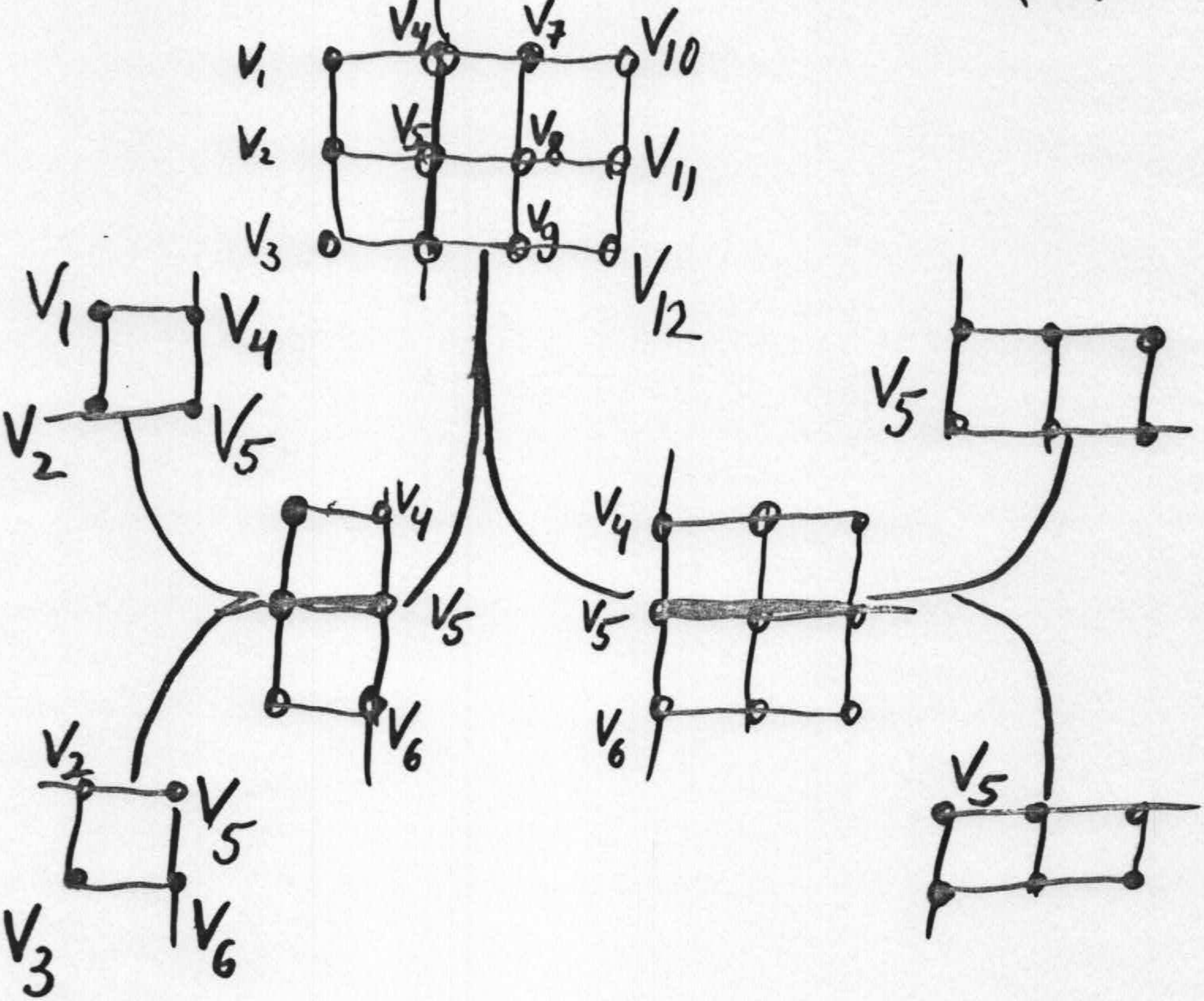
— Reorder variables of A so that variables in the group S_0 come first.

$$— \; A \xrightarrow[S_0]{\text{eliminate}} \overset{L_1^{-1}}{\begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ \underbrace{* \;\; *}_{S_0} & & & 1 & \\ & & & & \ddots \\ & & & & & 1 \end{bmatrix}} \begin{bmatrix} \mathcal{D}_1 & * \;\; * \;\; * \\ \hline 0 & A_1 \end{bmatrix} \quad (\beta)$$

$$— \; A_1 \xrightarrow[S_1]{\text{eliminate}} \quad L_2, \mathcal{D}_2, A_2,$$

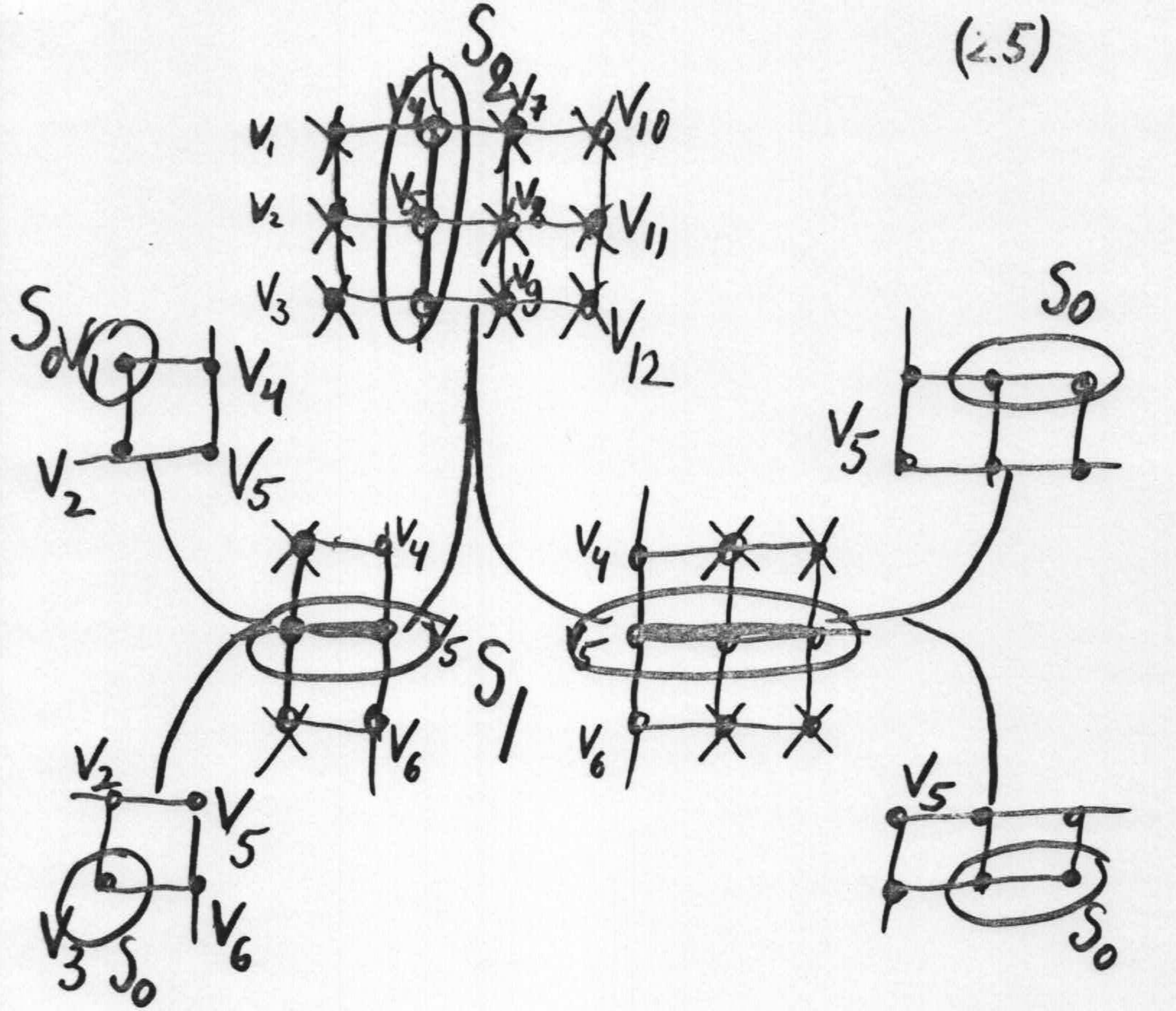
etc.

(2.5)



$O(\sqrt{n})$ - SEPARATOR TREE FOR GRAPH

(2.5)



TREE-NODE AT LEVEL i CONTAINS:

- SOME SUBSET OF S_i
- SUBSET OF $\bigcup_{k>i} S_k$, adjacent to S_i

FACTORING A^{-1} in parallel:

- Construct the separator tree with $d = \log n$ leve[ls]
- Split our matrix A into the ~~direct~~ "sum" of small matrices. Variables = vertices in corresponding tree-nodes on level 0.
- Inductive step: for tree level $i = 0, \ldots, d$:

1. For all small matrices in parallel, do partial G.E.
2. As a result, for each small matrix obtain L^{-1} and A_1 (see identity (β) above).
   All L^{-1} are left on the current level; all A_1 are moved to the next level.
3. Each A_1 is added up to its sibling in the separator tree

Claim: After we are done, sum of all small matrices L^{-1} from level i equals to L_i^{-1} (see β).

In other words: if variables are chosen according to the sep. tree then G.E and ⊕ commute with each other.

Handwaving proof: (i) fill-ins in G.E. correspond to certain paths in Graph(A).
(ii) Paths in Graph(A) "agree" with separator tree.

Rigorous proof is not very straightforward. I couldn't find any papers on this, so I wrote it down myself. Notes are available –

Lena.

MAIN IDEA (PAN-REIF)

— split the matrix into ($\oplus$ small matrices).

— PRIMITIVE OPERATIONS REQUIRED for factoring A^{-1} will commute with $\oplus$

⇒ We can apply these operations to small matrices on different levels of the separator tree.

- IN the END All factors of A^{-1} will be distributed among the tree-nodes.

Our Implementation:

- We use identity (α), with primitive operation = Gaussian elimination

They used different identity, with much more complicated primitive operation.

SPACE REQUIREMENTS:

1 MATRIX ELEMENT PER PROCESSOR

log(n) levels of the TREE

At EACH level, <u>SPACE = CONST • n</u>

FOR "NICE" GRID GRAPHS, CONST = 25

FOR "GENERAL" GRIDS, CONST ≈ 50

FOR PLANAR GRAPHS, MIGHT BE WORSE.

TIME REQUIREMENTS:

WORST CASE: LEVEL d-1 TIME = $O(\sqrt{n})$

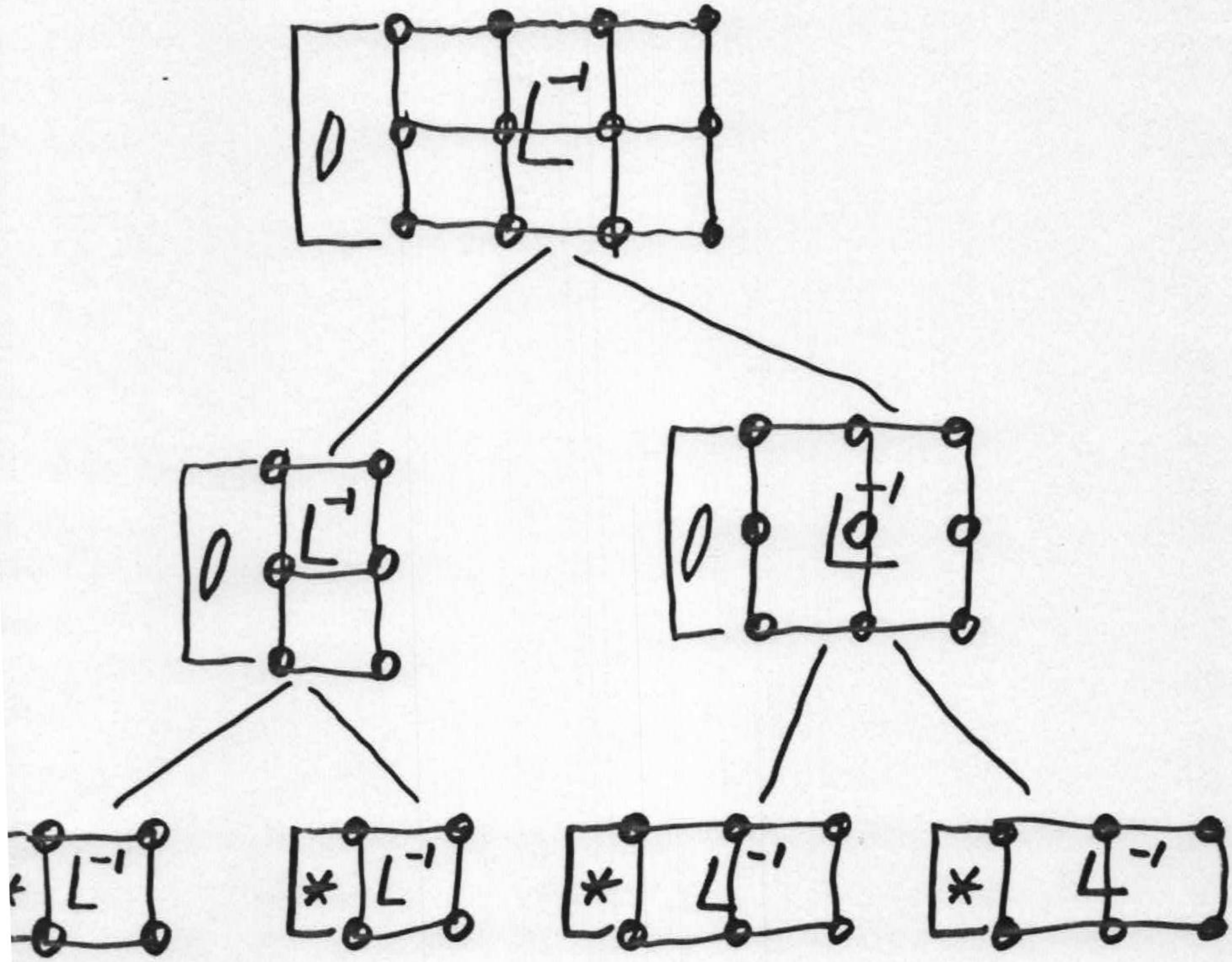
Backsolve

ALL FACTORS OF A^{-1} ARE NOW COMPUTED AND STORED IN THE CM.

SO, GIVEN vector b, we should be able to compute $A^{-1}b = x$.

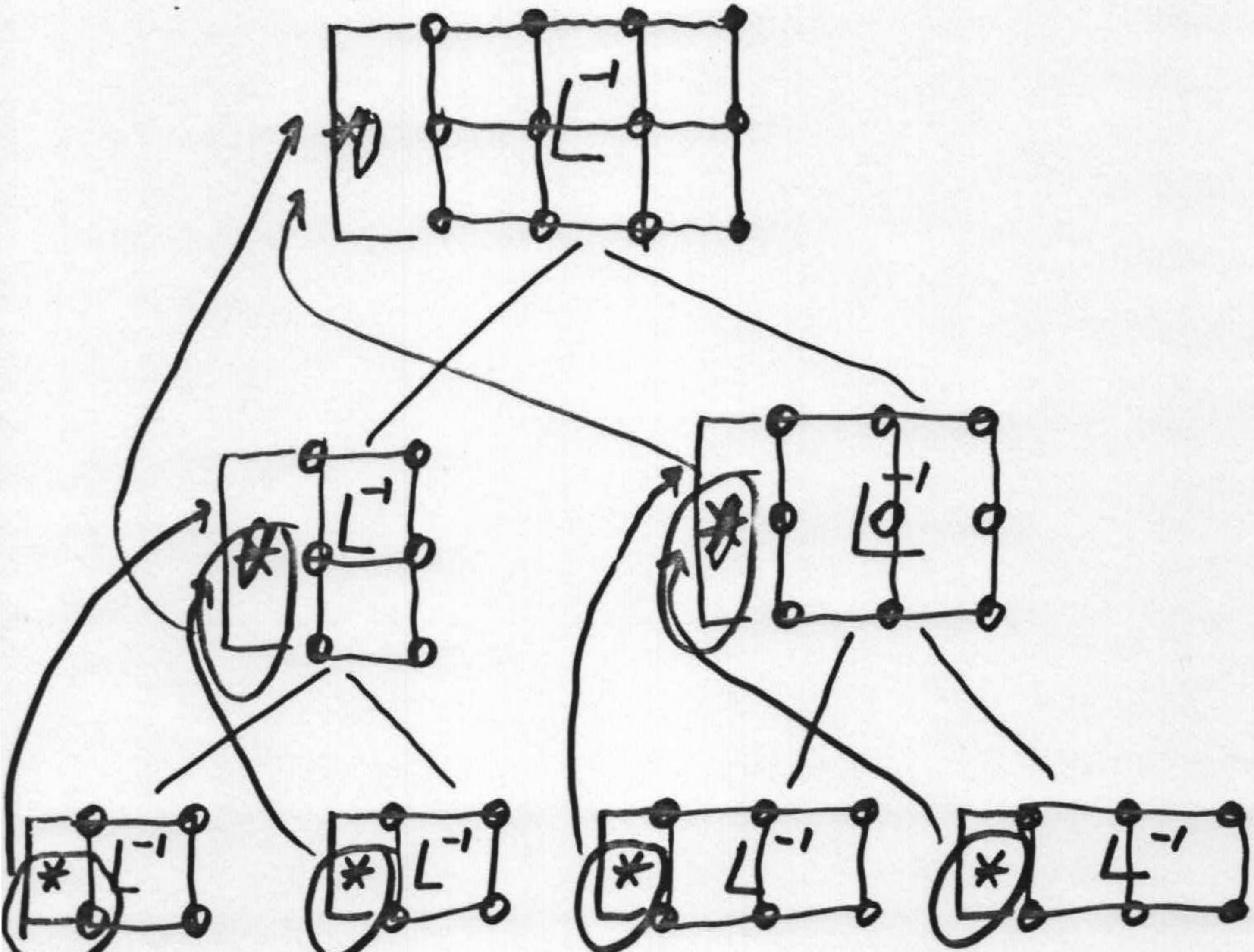
This computation is called backsolving.

BACKSOLVE: L^{-1} PART (GOING UP)



EACH MATRIX OCCUPIES A SQUARE IN CM AND CORRESPONDING PIECE OF VECTOR b OCCUPIES A COLUMN TO the left of IT.

BACKSOLVE: L^{-1} PART (GOING UP)



EACH MATRIX OCCUPIES A SQUARE IN CP
AND CORRESPONDING PIECE OF VECTOR b
OCCUPIES A COLUMN TO the left of IT.

BACKSOLVE: $U^{-1}D^{-1}$ PART
(GOING DOWN)

RUNNING TIME: BACKSOLVE:

$$\log(\sqrt{n}) + \log\left(\sqrt{\tfrac{n}{2}}\right) + \ldots \sim \underline{\text{const}\cdot\log n}$$

CM-Implementation-2:

DATA MAPPING.

IN LISPM: Construct the separator tree for Graph(A).

- Each tree-node at level i is a struct corresponding to some small submatrix of matrix A_ℓ (see (8)).
- slots of struct contain:
  - list of variables of this submatrix
  - its sublist, which we eliminate
  - address of one corner of this submatrix in the CM
  - a dozen of other things, later converted into pointers for the CM part of the algorithm

IN CM:

- allocate space for all submatrices
- create a structure for operating on a set of matrices. (by now we have a standard way of doing it)
- create pointers for sending matrix and vector elements from level to level.

To do this:

- First we sequentially load data from Lispm struct into lower left corners of matrices in the CM
- Then we spread data to all processors of the same matrix. This is done in parallel for all matrices.

Complete algorithm

(1) Create the necessary structures in LispM and CM
(= data mapping)

(2) Initialize submatrices and pieces of vector b at 0 level of CM, so that their direct sum equals A and b correspondingly.

(3) Factor A^{-1}

(4) Backsolve: going up and down.

(5) Read the solution x from CM
(The value of x is contained in the columns to the left of submatrices, where we initially put b.
Each processor knows to which variable it corresponds.
Processors, corresponding to the same variables, contain the same value of x).

Note: 2 and 5 are done sequentially.