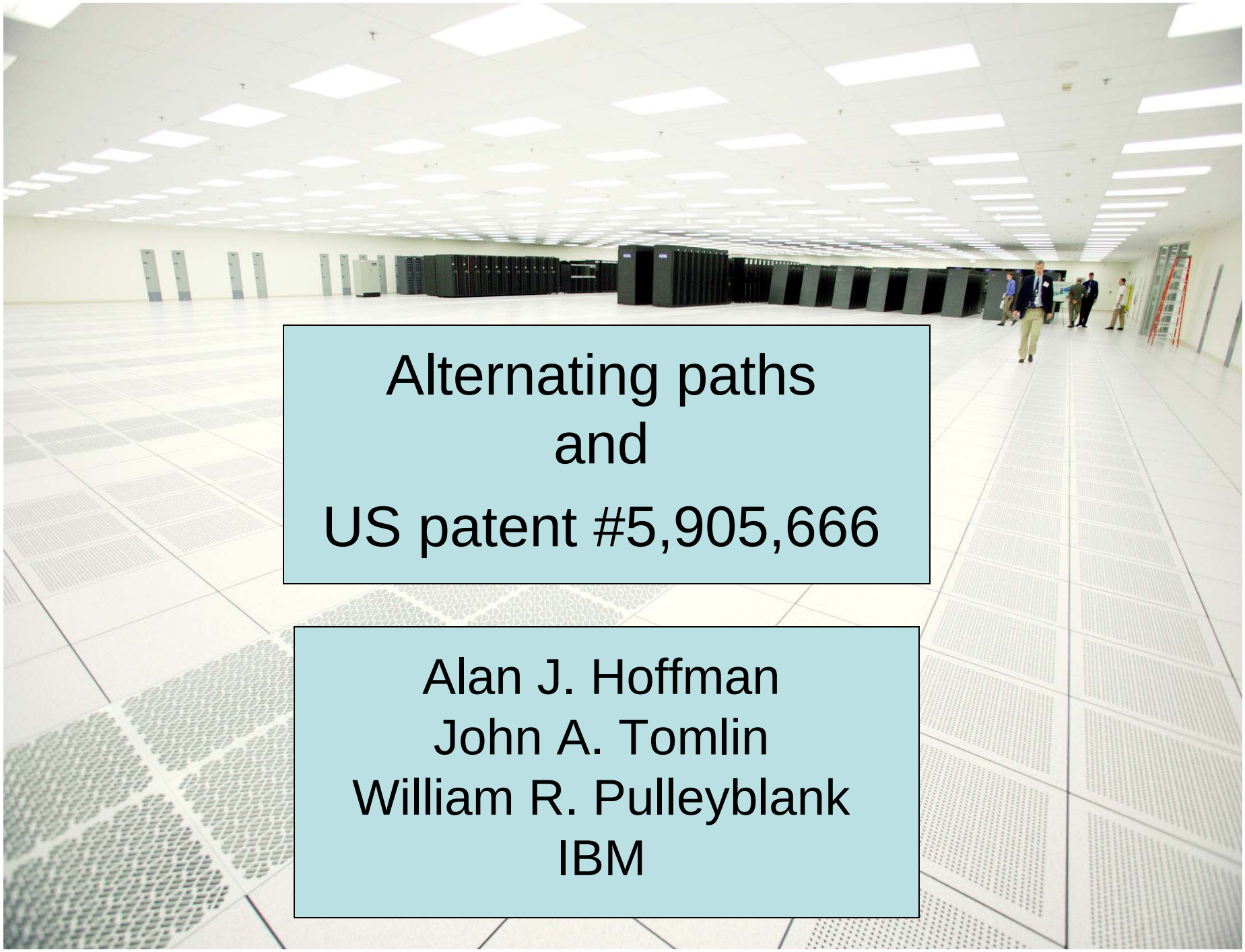


Alternating paths and US patent #5,905,666

Alan J. Hoffman
John A. Tomlin
William R. Pulleyblank
IBM

US Patent 5,905,666

PROCESSING SYSTEM AND METHOD FOR PERFORMING SPARSE MATRIX MULTIPLICATION BY REORDERING VECTOR BLOCKS

US005905666A

United States Patent [19]

Hoffman et al.

[11] **Patent Number: 5,905,666**

[45] **Date of Patent: May 18, 1999**

[54] **PROCESSING SYSTEM AND METHOD FOR PERFORMING SPARSE MATRIX MULTIPLICATION BY REORDERING VECTOR BLOCKS**

[75] Inventors: **Alan Jerome Hoffman**, Greenwich, Conn.; **William Robert Pulleyblank**, Cronton-on-Hudson, N.Y.; **John Anthony Tomlin**, Sunnyvale, Calif.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **08/931,901**

[22] Filed: **Aug. 28, 1997**

Related U.S. Application Data

[63] Continuation of application No. 08/367,627, Jan. 3, 1995, abandoned.

[51] **Int. Cl.**[6] **G06F 7/52**; G06F 19/00

[52] **U.S. Cl.** **364/754.02**; 364/468.05; 364/736.03; 364/841; 395/674; 705/8

[58] **Field of Search** 364/754.01, 754.02, 364/736.03, 468.05, 807, 822, 837, 841, 845; 395/674; 705/8

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,787,057	11/1988	Hammond	364/754.02
4,823,299	4/1989	Chang et al.	364/735
4,885,686	12/1989	Vanderbei	364/468.05
4,914,615	4/1990	Karmarkar et al.	364/754.02
4,918,639	4/1990	Schwarz et al.	364/757
5,021,987	6/1991	Chan et al.	364/754.02
5,025,407	6/1991	Gulley et al.	364/748.2
5,107,452	4/1992	Karmarkar et al.	364/754.02
5,170,370	12/1992	Lee et al.	364/736.03
5,185,715	2/1993	Zikan et al.	364/807
5,408,663	4/1995	Miller	395/674

FOREIGN PATENT DOCUMENTS

4037032A1	11/1990	Germany .
2272867	4/1989	Japan .

OTHER PUBLICATIONS

Kugel, Triple Multiplication of Sparse Matrices Technique, IBM Tech. Disclosure Bulletin, Y08710152, vol. 15, No. 2, pp. 376–377, Jul. 1972.

Gustavson, Efficient Algorithm to Perform Sparse Matrix Multiplication, IBM Tech. Disclosure Bulletin, Y08760479, vol. 20, No. 3, pp. 1262–1264, Aug. 1977.

Gustavson, Inverse of a Triangular Matrix Efficient Determination of the Boolean, IBM Tech. Disclosure Bulletin, vol. 23, No. 6, pp. 2614–2617, Nov. 1980.

Li and Sheng, Sparse Matrix Vector Multiplication of Polymorphic–Torus, IBM Tech. Disclosure Bulletin, vol. 32, No. 3A, Y08880066, pp. 233–238.

(List continued on next page.)

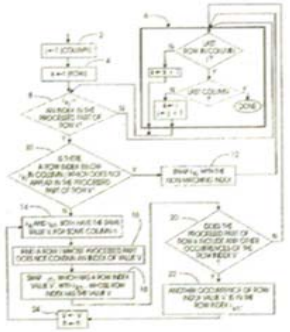
Primary Examiner—Emmanuel L. Moise

Attorney, Agent, or Firm—James C. Pintner

[57] **ABSTRACT**

A method, system, and data structure are provided which facilitate matrix multiplication with advantageous computational efficiency. The invention, as variously implemented as a processing system, method, or data structure in a recording medium such as a memory, has applicability to numerous fields, including linear programming, where a great deal of multiplication of large, sparse matrices is performed. The method of the invention includes the steps of creating a first submatrix block from non-zero terms of a sparse matrix, such that all of the terms within a given column of the submatrix block are form a respective column of the sparse matrix, creating a corresponding second index submatrix block of the same dimensions as the first block, such that each term of the second block identifies the position of the corresponding term of the first block within the sparse matrix, in terms of a row and column index. Finally, the method includes reordering terms of the first and second blocks correspondingly, as necessary to produce a final configuration within the first and second blocks such that all of the row indices within any given row of the second block are distinct.

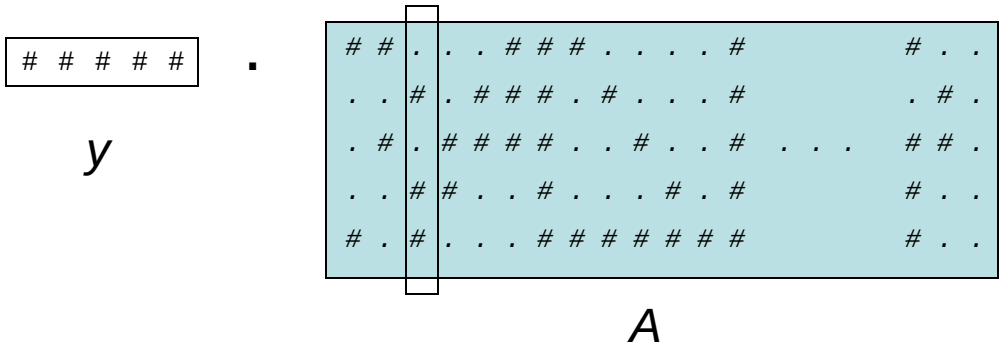
12 Claims, 7 Drawing Sheets



The problem: Efficiently computing *Ax* and *yA*

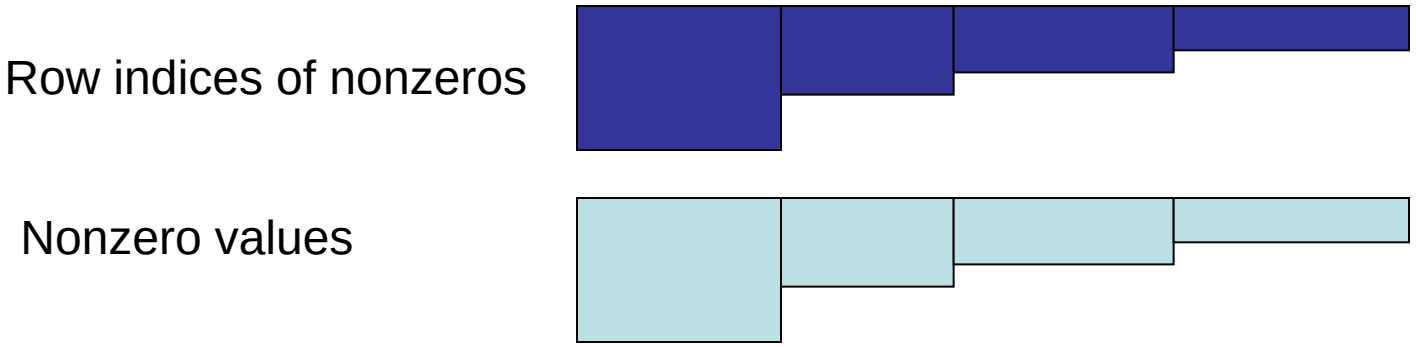
- Linear programming:
  - Primal: max cx, $Ax = b, x \geq 0$
  - Dual: min yb, $yA \geq c$
  - Requires computation of *Ax* and *yA*
- Simplex algorithm:
  - Small number of computations of *Ax,* but many computations of *yA* when performing pricing
    - Critical to performance of algorithm
- Interior algorithms
  - Many fewer iterations, but require similar number of computations of *Ax* and *yA*

Computational details - *yA*

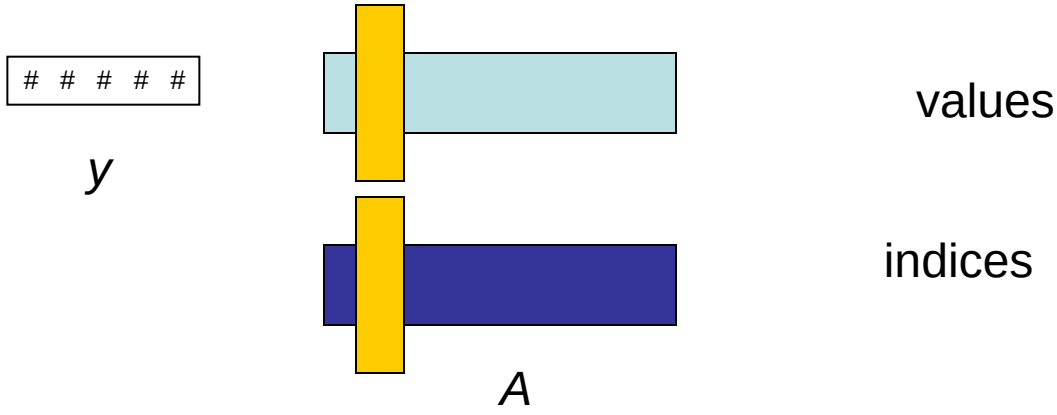


Problem: typically the matrices are sparse
in practice, typically 4 to 8 nonzeros per column

Partition matrix based on the number of nonzeros per column



Now we can focus on dense submatrices



Normal (scalar) computation:

18 machine cycles per calculation; 15 cycles initialization

Vector Unit computation:

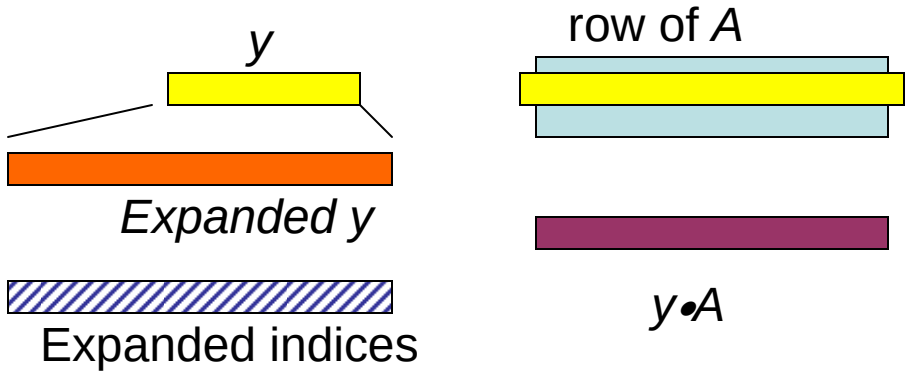
180 machine cycles to startup, 4 cycles per calculation

- Require approximately 12 elements to break even

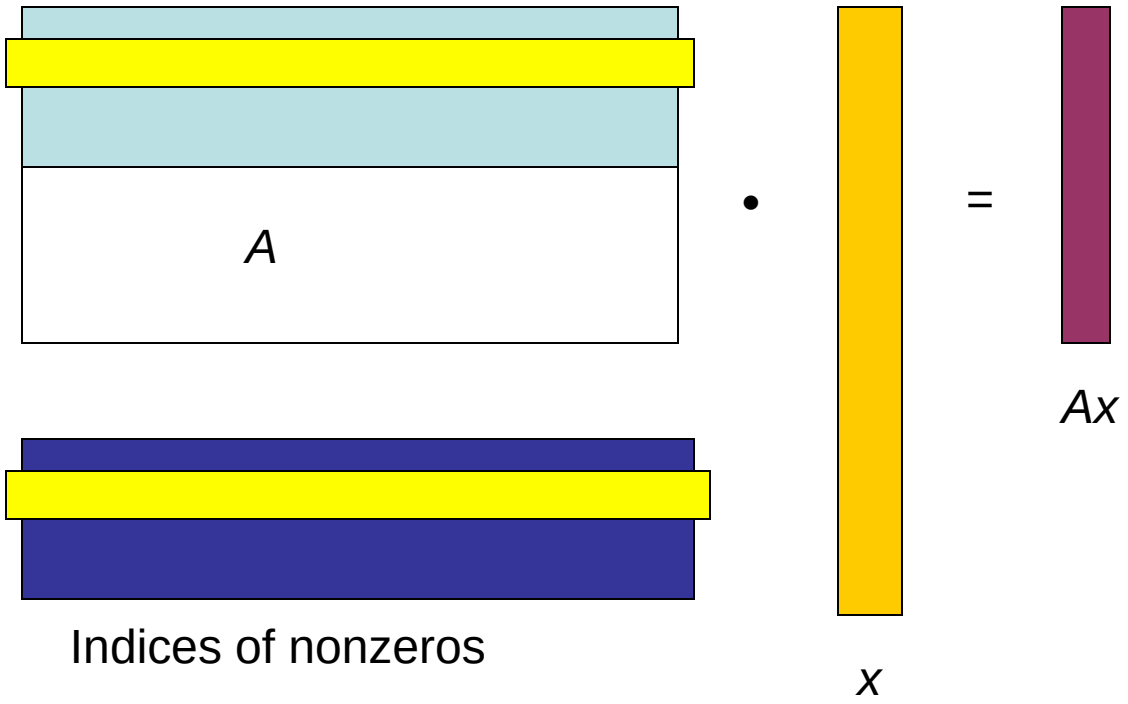
- so, do computation by rows

- Use indices to select components of *y*

-Use "accumulator" to compute $y \bullet A_j$



What goes wrong with the computation of *Ax*?



Entries in index rows now specify which component of *Ax* is being computed.
If all entries in each index row are distinct, then it works.

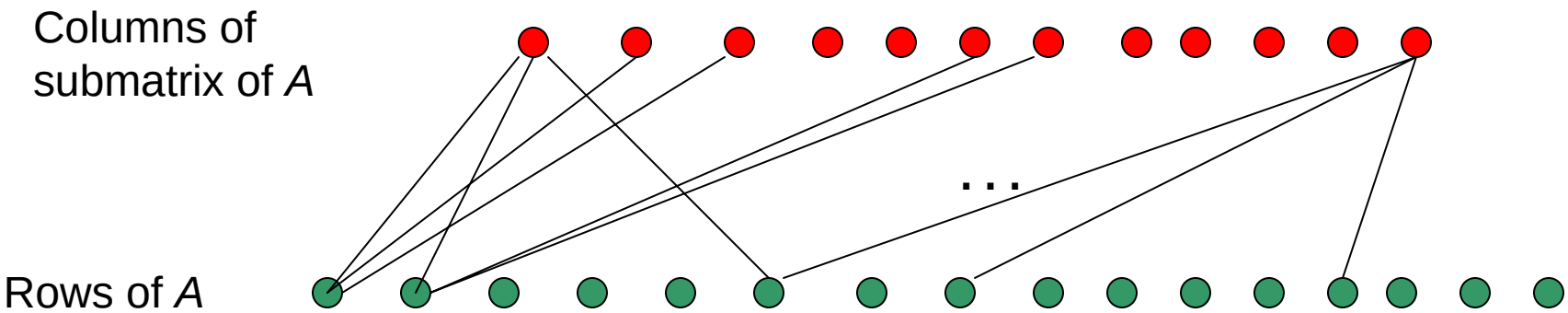
If there are duplicate entries, then we can attempt to permute the entries in each column to eliminate duplicates.

Example

Suppose *A* has 16 rows, and our block has 12 columns and 3 rows.

Index matrix

1	3	1	16	15	2	2	14	7	5	7	6
2	4	5	15	11	8	9	4	11	3	3	8
6	1	6	4	9	7	14	12	10	11	8	13



Bipartite graph representation

Applying Koenig's edge coloring theorem

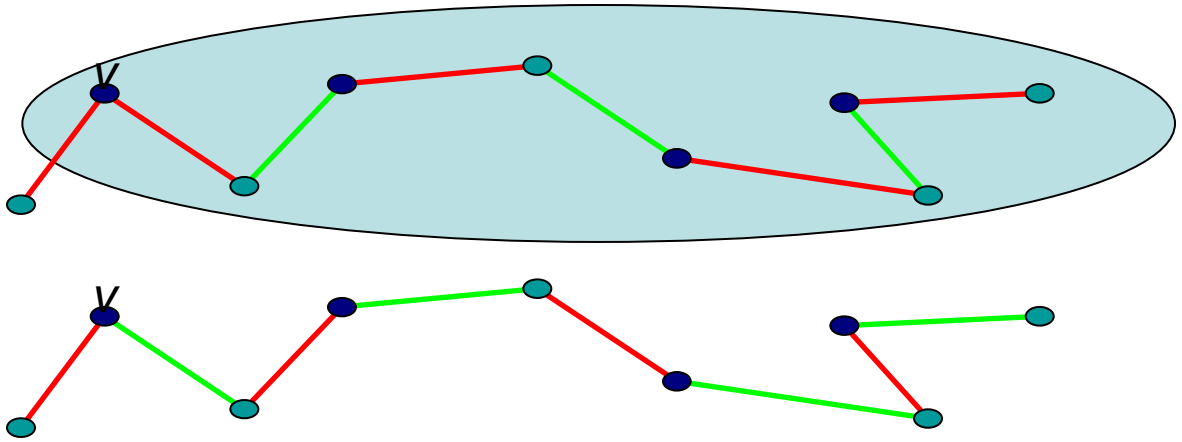
- Let *d* be the number of rows in the submatrix. Permuting the entries in each column, so that no row contains duplicates, is equivalent to edge coloring the bipartite graph with *d* colors.

- By Koenig's theorem, this is possible if and only if each node of the bipartite graph has degree at most *d*, which is equivalent to each index appearing at most *d* times in the submatrix.

- We need an efficient way to obtain the edge coloring. If there is no coloring, because some index occurs too often, we can permute these to the last row and reduce the size of the block.

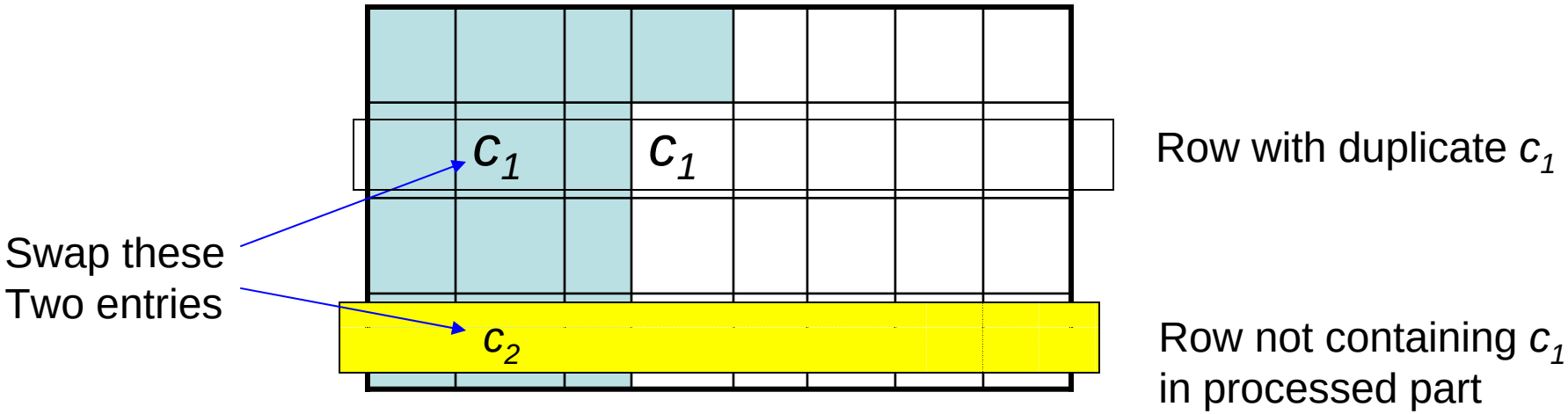
Alternating path based proof

- Faber, Ehrenfeucht and Kierstead (1982) presented an alternating path based proof:
  - Start with any coloring of the edges.
  - If some node *v* is incident with two identically colored edges c_1, then some edge color c_2 must be missing
  - Switch the edges in a c_1-c_2 alternating path'



Matrix interpretation

- Start with any permutations in the columns of the submatrix; process columns left to right, top to bottom



If this creates a c_2 conflict with processed elements, it involves original row, repeat.

Computational results

Model	Rows	Columns	Nonzeros	Average # of nonzeros per column
BandM	185	340	1670	4.9
Degen2	381	473	3580	7.5
25FV47	715	1484	9994	6.7
PILOTS	1374	3361	40757	12.1
Degen3	1412	1727	24465	14.2
Mod41	34952	94752	210211	2.2
Mod42	40022	110475	244439	2.2
MPS4	90482	219249	502943	2.3

Computational results

Model	No. of *Ax* calcs.	Old *Ax* time	Reordering time	New *Ax* time	No. of *yA* calcs.	Total *yA* time
BandM	79	0.056	0.007	0.043	46	0.018
Degen2	68	0.101	0.020	0.062	40	0.028
25FV47	134	0.486	0.044	0.416	79	0.099
PILOTS	184	2.440	0.216	1.662	109	0.523
Degen3	93	0.676	0.089	0.462	53	0.186
Mod41	213	17.672	1.340	11.209	127	3.793
Mod42	278	26.746	1.289	17.169	166	5.863
MPS4	408	82.826	3.361	43.543	244	18.187

Reference

A.J. Hoffman, W.R. Pulleyblank, J.A. Tomlin, On computing *Ax* and $\pi^T A$, when *A* is sparse, *Annals of Numer. Math.* 4 (1997) 359-367.