A Note on the Performance of Sparse Matrix-vector Multiplication with Column Reordering

Sardar Anisul Haque
University of Western Ontario, Ontario, Canada
Shahadat Hossain
University of Lethbridge, Alberta, Canada

June 25, 2009

Principle of Locality

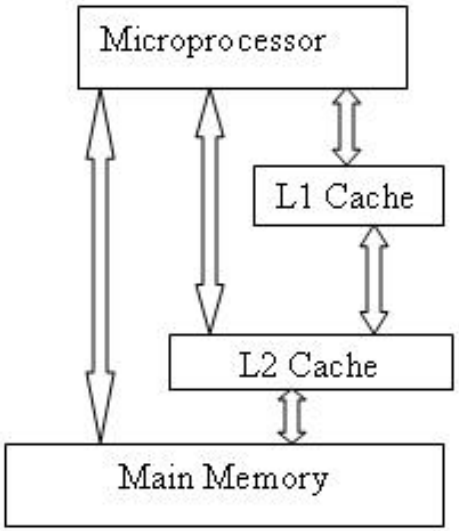
The principle of locality states that most programs do not access their code and data uniformly. There are mainly two types of locality:

1. Spatial locality: It refers to the observation that most programs tend to access data sequentially.
2. Temporal locality: It refers to the observation that most programs tend to access data that was accessed previously.

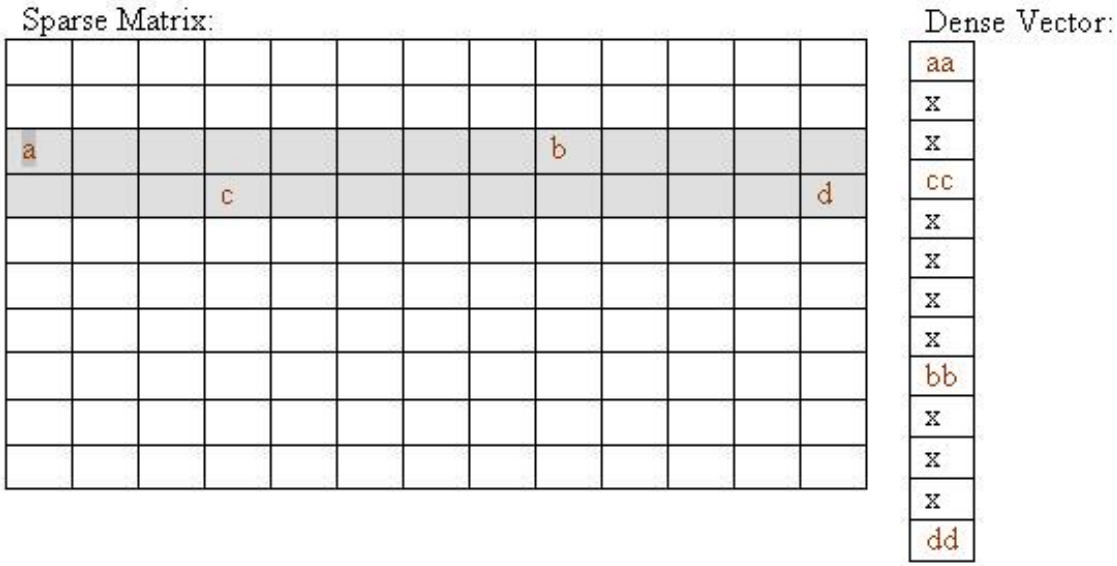
Performance gap between CPU speed and Main memory speed

- CPU speed improvement: 35% to 55% (in a year).
- Main memory latency improvement: 7% (in a year).

Hierarchical Memory Systems



Data Locality in sparse matrix-vector multiplication



Computing $y = Ax$ on modern superscalar architecture often exhibits

- Poor data locality.
- Large volume of load operations from memory compared to the floating point operations.
- Indirect access to the data.
- Loop overhead.

Improving Data Locality of x in computing $y = Ax$

Preprocess A by permuting the rows or columns of A in such a way that

- the number of nonzero block is reduced to improve the spatial locality of x.
- the nonzeros of each column are consecutive to improve the temporal locality of x.

But this preprocessing phase can be computationally expensive.

Conjugate Gradient Algorithm

- An iterative method to obtain numerical solution of large system of linear equations $Ax = b$.
- In this method, A remains unchanged and we need to multiply it with a vector.
- The method may require a good number of iterations before convergence.

Yousef Saad. *Iterative methods for sparse linear systems, 2nd Edition*. SIAM, 2003.

Storage schemes for sparse matrices

The names of some well known storage schemes for sparse matrices are given below.

- Compressed Row Storage (CRS) scheme.
- Fixed-size Block Storage (FSB) Scheme.
- Block Compressed Row Storage (BCRS) scheme.

FSB Scheme

We define a *nonzero block* as a sequence of $k \geq 1$ contiguous nonzero elements in a row.
We will denote this storage scheme by *FSBl*, where the last character *l* represents the length of the nonzero block. For example, *FSB2* represents fixed-size block storage scheme of length 2. In FSBl scheme the given sparse matrix A is expressed as a sum of two matrices A_1 and A_2; A_1 stores all the nonzero block of size *l* and A_2 stores the rest (in *CRS* scheme).

S. Toledo. Improving Memory-System Performance of Sparse Matrix-Vector Multiplication. In *Proceedings of the 8th SIAM Conference on Parallel Processing for Scientific Computing*, 1997.

$A = A_1 + A_2$ **considering** $l = 2$

$$A = \begin{pmatrix} 0 & 0 & a_{02} & a_{03} & 0 \\ a_{10} & 0 & 0 & 0 & a_{14} \\ 0 & a_{21} & 0 & a_{23} & 0 \\ a_{30} & 0 & a_{32} & 0 & a_{34} \\ 0 & 0 & a_{42} & a_{43} & 0 \end{pmatrix}$$

$$A_1 = \begin{pmatrix} 0 & 0 & a_{02} & a_{03} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{42} & a_{43} & 0 \end{pmatrix} \qquad A_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ a_{10} & 0 & 0 & 0 & a_{14} \\ 0 & a_{21} & 0 & a_{23} & 0 \\ a_{30} & 0 & a_{32} & 0 & a_{34} \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$A_1 = \begin{pmatrix} 0 & 0 & a_{02} & a_{03} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{42} & a_{43} & 0 \end{pmatrix}$$

FSB2 data structure for sparse matrix A_1

valueBl	a_{02}	a_{03}	a_{42}	a_{43}		
cloindBl	2	2				
rowptrBl	0	1	1	1	1	2

Algorithm Sparse matrix vector multiplication for FSB2

```
1:  for i ← 0 to m − 1 do
2:      for k ← rowptrBl[i] to rowptrBl[i+1] − 1 do
3:          j ← colindBl[k]
4:          l ← 2 ∗ k
5:          y[i] += valueBl[l] ∗ x[j]
6:          y[i] += valueBl[l+1] ∗ x[j+1]
7:      end for
8:      for k ← rowptr[i] to rowptr[i+1] − 1 do
9:          j ← colind[k]
10:         y[i] += value[k] ∗ x[j]
11:     end for
12: end for
```

Column ordering problem

We define *column ordering problem* as follows.
Given an $m \times n$ sparse matrix A, find a permutation of columns that minimizes β, where β is the total number of nonzero blocks in A.

Ali Pinar and Michael T. Heath. Improving performance of sparse matrix-vector multiplication. In *Supercomputing '99: Proceedings of the 1999 ACM/IEEE conference on Supercomputing (CDROM)*, New York, NY, USA, 1999. ACM.

Weight of intersection

Columns j and l of matrix A are said to *intersect* if there is a row i such that $a_{ij} \neq 0$ and $a_{il} \neq 0$. The *weight of intersection* of any two columns j and l, denoted by w_{jl}, is the number of rows in which they intersect.

Column ordering algorithms

The names of some column ordering algorithms are given below.

- Column intersection ordering.
- Similarity ordering.
- Local Improvement ordering.
- Binary reflected gray code ordering.

Column intersection ordering algorithm

$$\begin{pmatrix} 0 & 0 & 0 & a_{03} & 0 \\ a_{10} & 0 & 0 & 0 & a_{14} \\ 0 & a_{21} & a_{22} & a_{23} & 0 \\ a_{30} & 0 & 0 & 0 & a_{34} \\ 0 & 0 & 0 & a_{43} & 0 \\ a_{50} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{63} & a_{64} \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 0 & a_{03} & 0 & 0 \\ a_{10} & a_{14} & 0 & 0 & 0 \\ 0 & 0 & a_{23} & a_{21} & a_{22} \\ a_{30} & a_{34} & 0 & 0 & 0 \\ 0 & 0 & a_{43} & 0 & 0 \\ a_{50} & 0 & 0 & 0 & 0 \\ 0 & a_{64} & a_{63} & 0 & 0 \end{pmatrix}$$

Similarity ordering algorithm

In this column ordering algorithm, the weight of intersection between two columns i and j is the number of rows in which both of them have either zero or nonzero.

Similarity ordering algorithm (contd..)

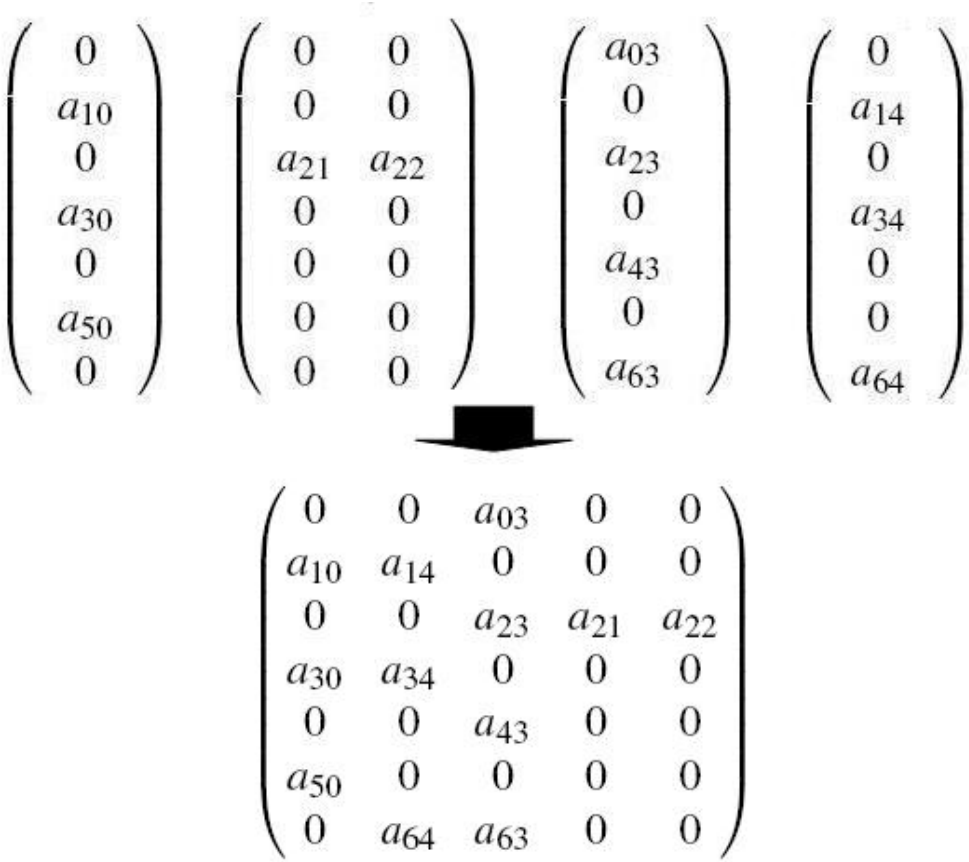
$$\begin{pmatrix} 0 & 0 & 0 & a_{03} & 0 \\ a_{10} & 0 & 0 & 0 & a_{14} \\ 0 & a_{21} & a_{22} & a_{23} & 0 \\ a_{30} & 0 & 0 & 0 & a_{34} \\ 0 & 0 & 0 & a_{43} & 0 \\ a_{50} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{63} & a_{64} \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 0 & a_{03} & 0 & 0 \\ 0 & 0 & 0 & a_{14} & a_{10} \\ a_{21} & a_{22} & a_{23} & 0 & 0 \\ 0 & 0 & 0 & a_{34} & a_{30} \\ 0 & 0 & a_{43} & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{50} \\ 0 & 0 & a_{63} & a_{64} & 0 \end{pmatrix}$$

Local Improvement ordering algorithm

$$\begin{pmatrix} 0 & 0 & 0 & a_{03} & 0 \\ a_{10} & 0 & 0 & 0 & a_{14} \\ 0 & a_{21} & a_{22} & a_{23} & 0 \\ a_{30} & 0 & 0 & 0 & a_{34} \\ 0 & 0 & 0 & a_{43} & 0 \\ a_{50} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{63} & a_{64} \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ a_{10} \\ 0 \\ a_{30} \\ 0 \\ a_{50} \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ a_{21} & a_{22} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \quad \begin{pmatrix} a_{03} \\ 0 \\ a_{23} \\ 0 \\ a_{43} \\ 0 \\ a_{63} \end{pmatrix} \quad \begin{pmatrix} 0 \\ a_{14} \\ 0 \\ a_{34} \\ 0 \\ 0 \\ a_{64} \end{pmatrix}$$

Local Improvement ordering algorithm (contd..)



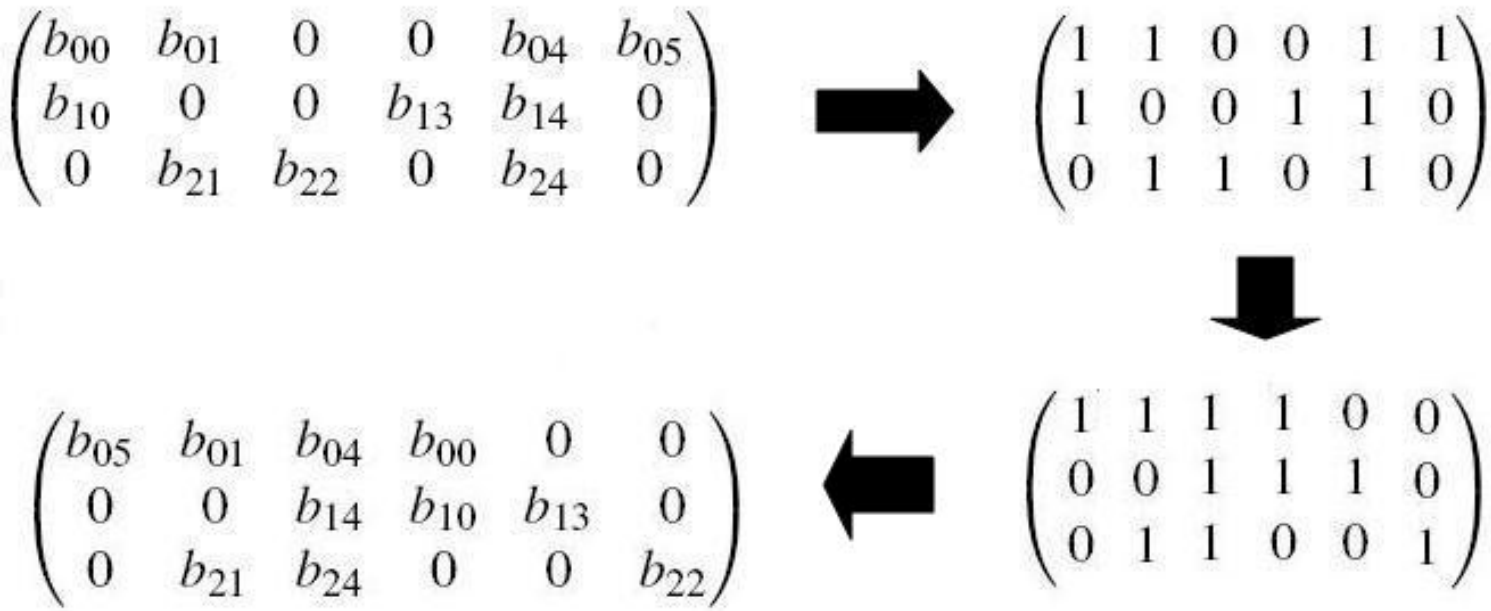
Binary reflected gray code ordering

The main scientific contribution of this thesis is as follows:
We propose column ordering algorithm based on *binary reflected gray code* for sparse matrices. To the best of our knowledge we are the first to consider gray codes for column ordering in sparse matrix-vector multiplication. We call it *binary reflected gray code ordering* or BRGC algorithm.

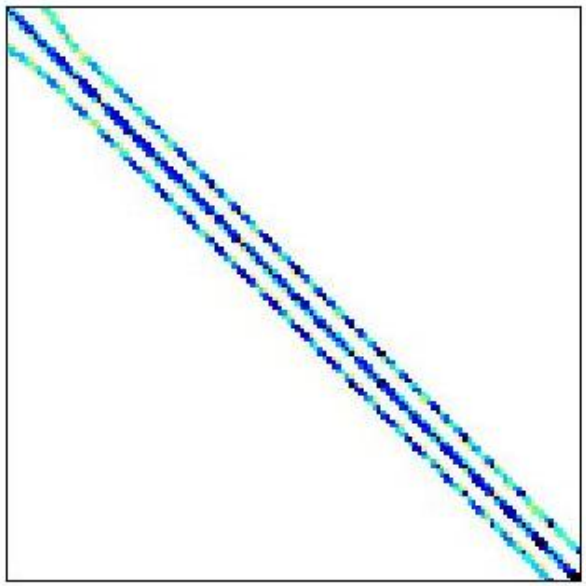
Binary reflected gray code

$$G^p = [0G_0^{p-1}, \ldots, 0G_{2^{p-1}-1}^{p-1}, 1G_{2^{p-1}-1}^{p-1}, \ldots, 1G_0^{p-1}]$$
$$G^3 = [000, 001, 011, 010, 110, 111, 101, 100]$$

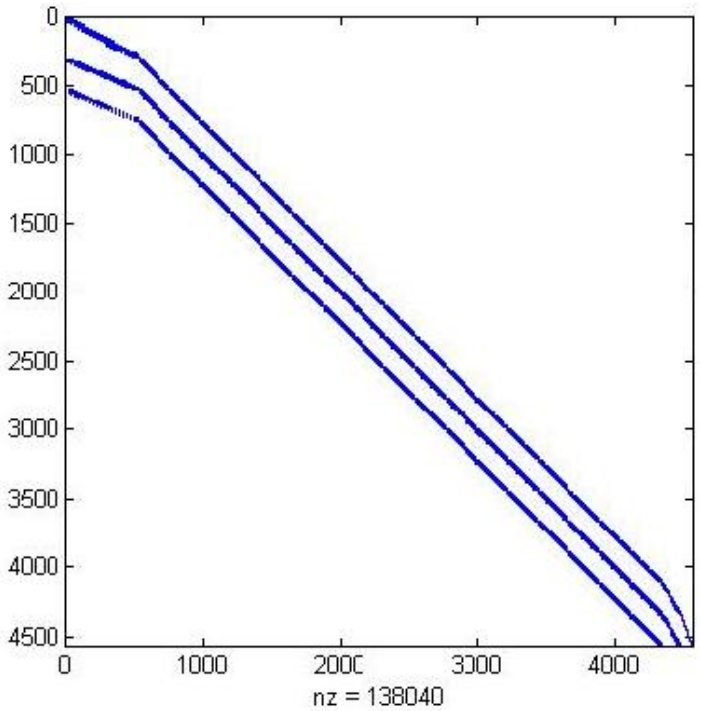
Binary reflected gray code ordering algorithm (BRGC) (contd..)



Example: cavity26

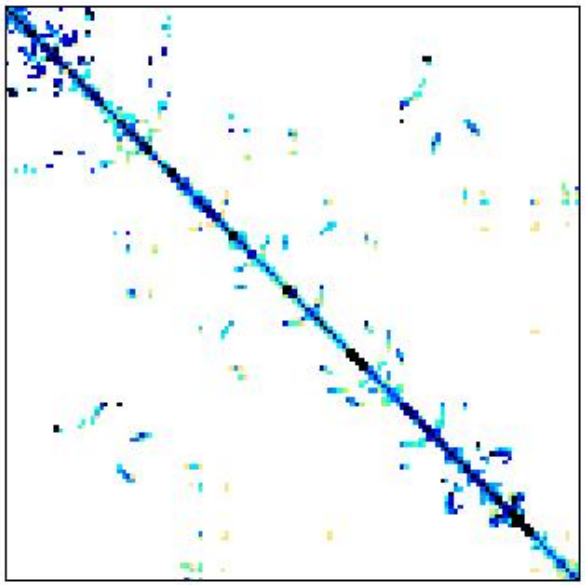


(a)

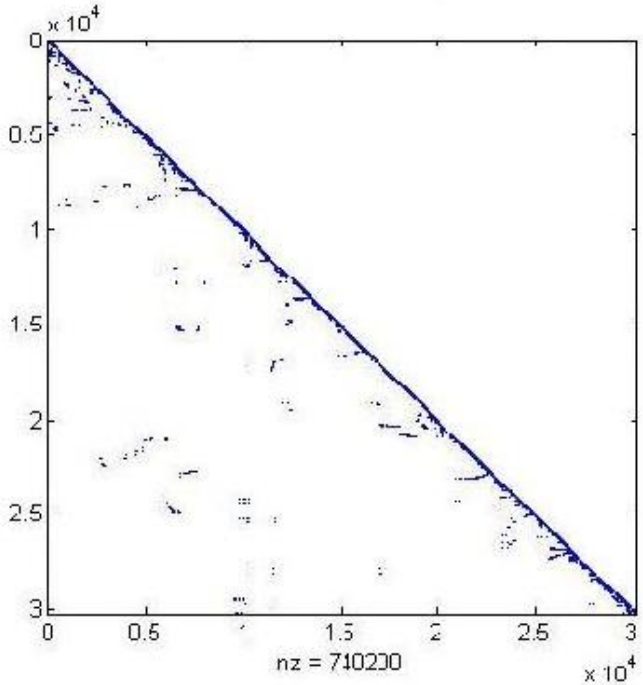


(b)

Example: bcsstk35



(a)



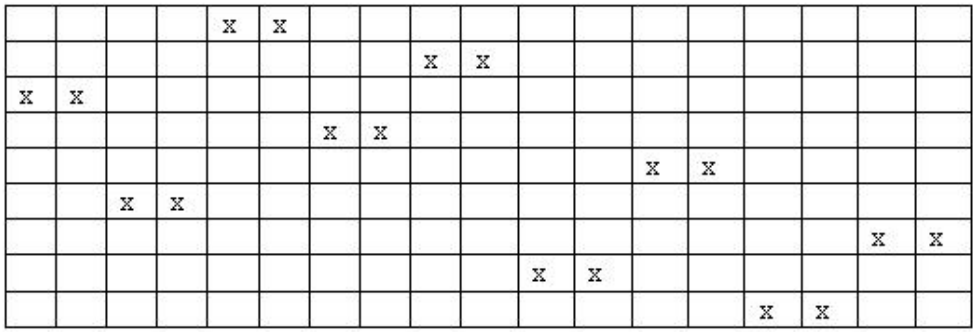
(b)

Data locality and column ordering algorithms

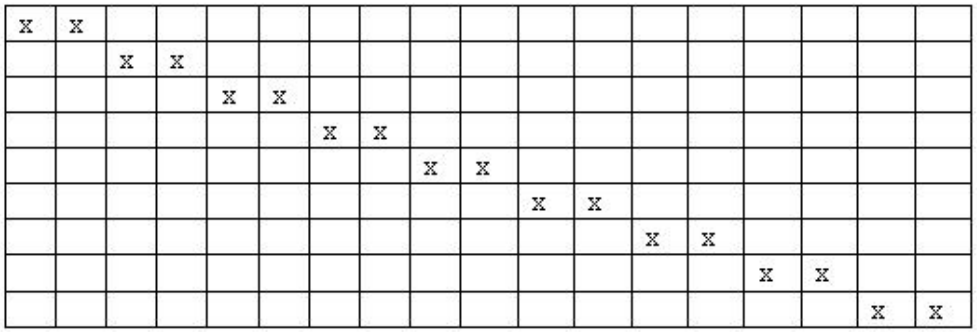
Let π is the column permutation found by *column intersection ordering* or *local improvement ordering* or *similarity ordering* algorithm. Here column $\pi[i+1]$ is found by looking at the nonzeros of column $\pi[i]$. But the data locality of A should be evaluated over more than pairs of columns.

D. B. Heras, J. C. Cabaleiro, and F. F. Rivera. Modeling data locality for the sparse matrix-vector product using distance measures. *Journal of Parallel Computing*, 27:897–912, 2001.

Data locality and column ordering algorithms (contd..)



Data locality and column ordering algorithms (contd..)



Features of BRGC ordering algorithm

- It improves both temporal and spatial locality of x in computing $y = Ax$.
- The given column ordering of input matrix has no effect on it.
- It does not change the sparsity structure of a banded matrix much.

Table: Computing platforms

Name	Compaq	ibm	sun
Processor name	AMD Athlon(tm)64 3500+	Intel pentium4	Ultra sparc-IIe
Processor Speed	2.2 GHz	2.8 GHz	550 MHz
RAM	512 MB	1 GB	384 MB
OS	Linux	Linux	Sun Solaries
L2 Cache	512 KB	512 KB	256 KB
L2 Cache type	16-way set associative	8-way set associative	8-way set associative and direct mapped
L2 Cache line size	64 bytes	64 bytes	64 bytes

Input matrices

26 matrices from linear programming problem, structural problem, optimization problem, economic problem, circuit simulation problem etc.
Source: Tim Davis, University of Florida Sparse Matrix Collection, url: http: www.cise.ufl.edu/research/sparse. Access Date: April 10, 2008.

Performance measure

- We use CPU time (for example $t_{A,SpMxV(compaq,crs,O_{brgc})}$) as performance measure.

Performance ratio

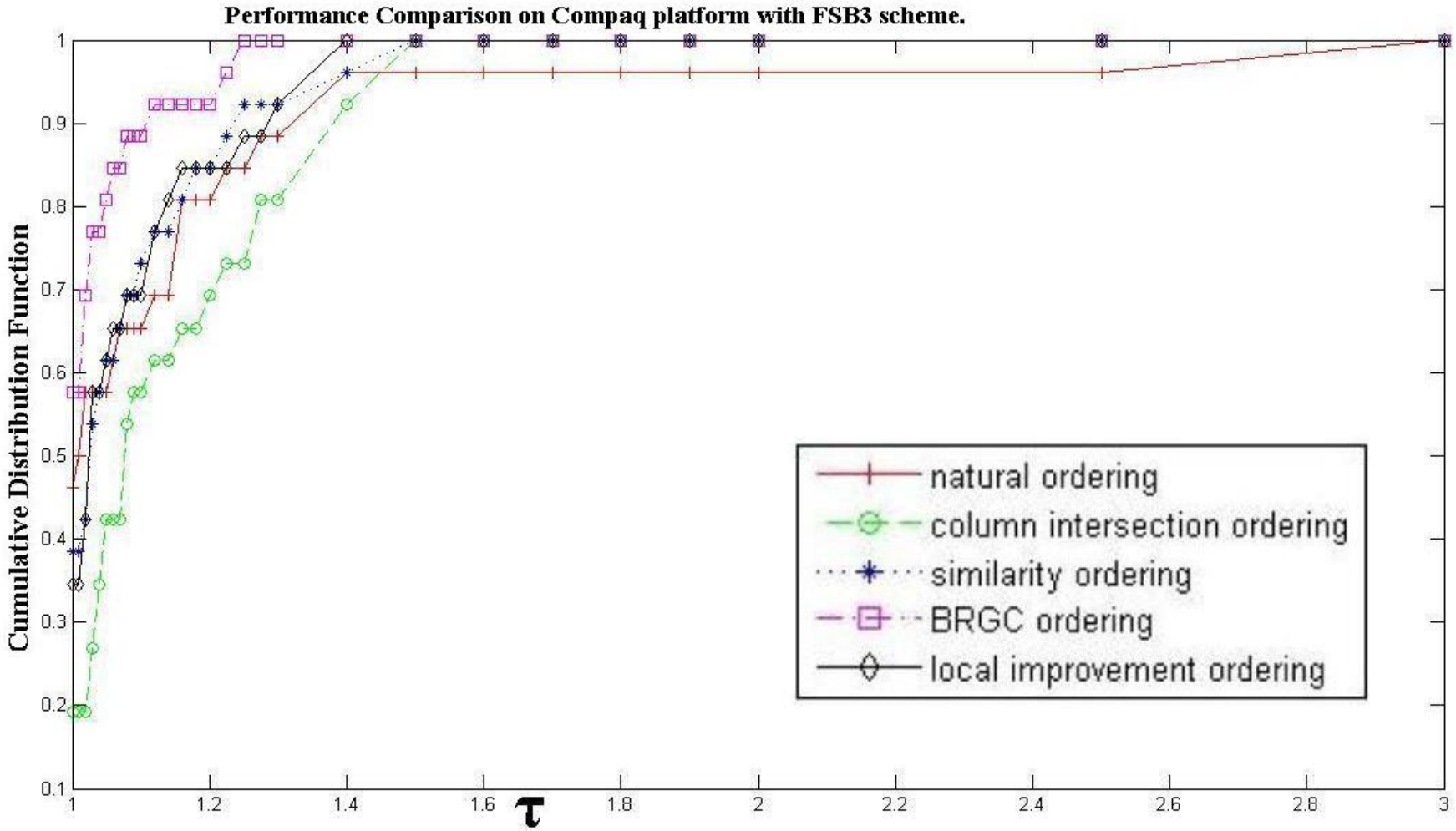
We define performance ratio as

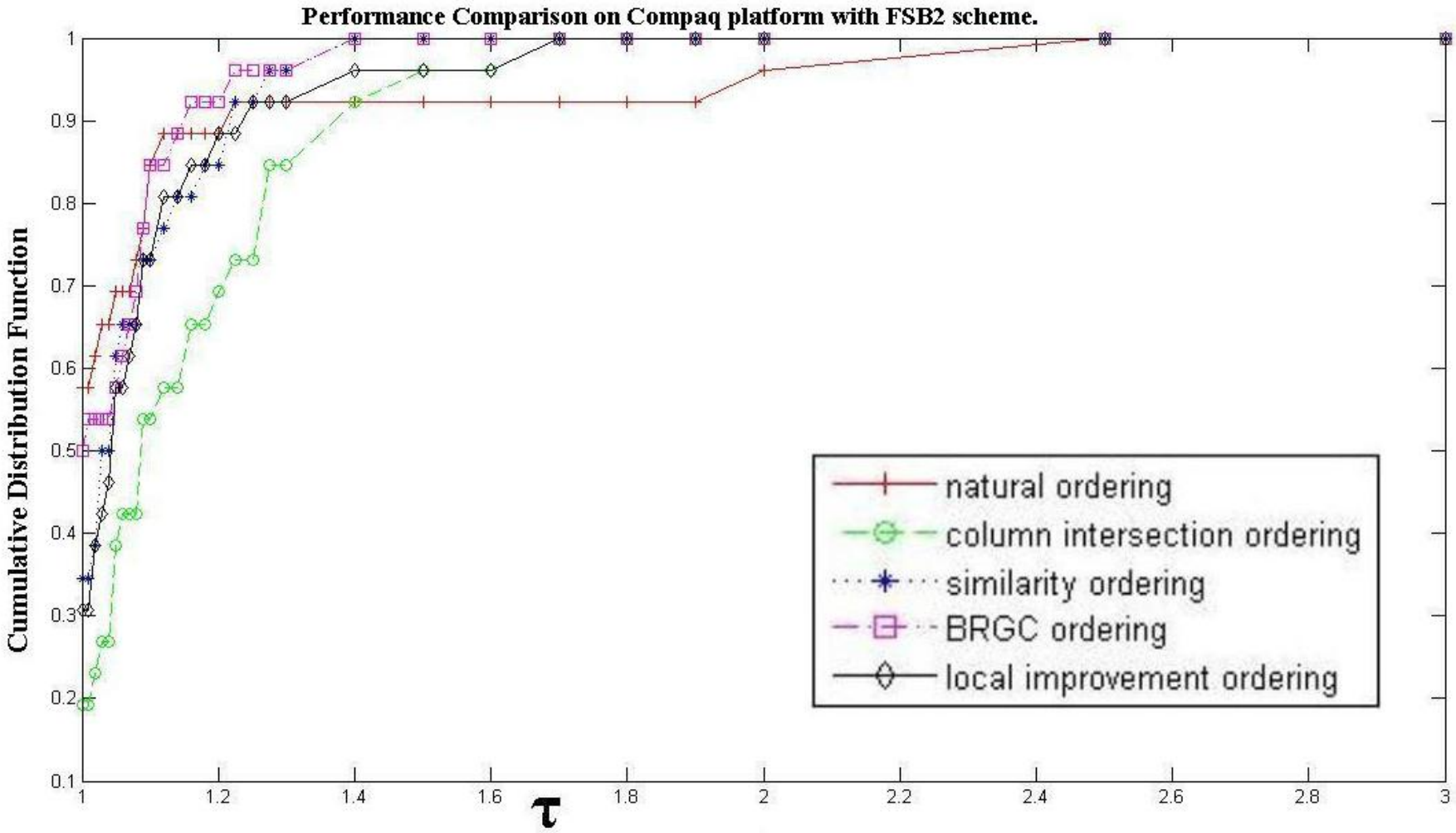
$$r_{A,SpMxV(pl,ss,ra)} = \frac{t_{A,SpMxV(pl,ss,ra)}}{min\{t_{A,SpMxV(pl,ss,ANY)}\}}$$

Evaluation method

Finally, the performance of a $SpMxV(pl, ss, ra)$ can be measured by the following cumulative distribution function: $\rho_{SpMxV(pl,ss,ra)}(\tau) = \frac{1}{|\Gamma|} size\{A \in \Gamma : r_{A,SpMxV(pl,ss,ra)} \leq \tau\}$, where, Γ is the set of input matrices.

Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.


Performance Comparison on Compaq platform with FSB3 scheme.
Cumulative Distribution Function
τ
natural ordering
column intersection ordering
similarity ordering
BRGC ordering
local improvement ordering


Performance Comparison on Compaq platform with FSB2 scheme.
Cumulative Distribution Function
τ
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
3
natural ordering
column intersection ordering
similarity ordering
BRGC ordering
local improvement ordering

Conclusion

- If the distribution of nonzeros of a sparse matrix is very much sparse or the number of nonzero blocks is very high then permuting the rows or columns of that sparse matrix is necessary.
- Fixed-size block storage scheme performs better than CRS and BCRS schemes.
- We found BRGC ordering is competitive with other column ordering algorithms during sparse matrix-vector multiplication.

Future direction

- Applicability of BRGC ordering to other sparse matrix problems requires further investigation.
- Use of register blocking and cache blocking method in sparse matrix-vector multiplication in addition to BRGC ordering.
- Applying BRGC ordering in fixed size blocking storage schemes (both rows and columns) of sparse matrices.

Thank you