

# Sequential Sparse Matrix–Vector Multiplication

Section 4.1 of Parallel Scientific Computation, 2nd edition

Rob H. Bisseling

Utrecht University

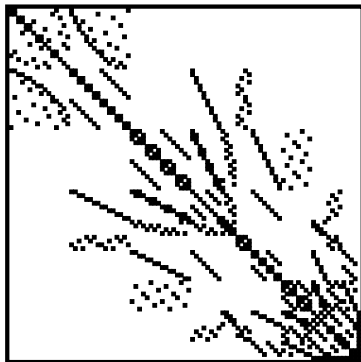


# Sparse and dense matrices

- ▶ **Sparse matrices** are sparsely populated by nonzero elements.
- ▶ **Dense matrices** are densely populated by nonzeros.
- ▶ Sparse matrix computations **save time**: operations with zeros can be skipped or simplified; only the nonzeros must be handled.
- ▶ Sparse matrix computations also **save memory**: only the nonzeros need to be stored (together with their location).



## Sparse matrix cage6



- ▶  $n = 93$  rows/columns
- ▶  $nz = 785$  nonzeros
- ▶  $c = 8.4$  nonzeros per row
- ▶  $d = 9.1\%$  density



# Matrix statistics

- ▶ Number of nonzeros is

$$nz = nz(A) = |\{a_{ij} : 0 \leq i, j < n \wedge a_{ij} \neq 0\}|.$$

- ▶ Average number of nonzeros per row or column is

$$c = c(A) = \frac{nz(A)}{n}.$$

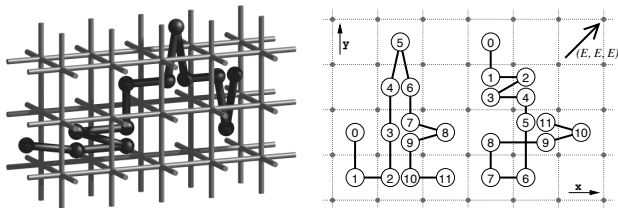
- ▶ Density is

$$d = d(A) = \frac{nz(A)}{n^2}.$$

- ▶ Matrix is **sparse** if  $nz(A) \ll n^2$ , or  $c(A) \ll n$ , or  $d(A) \ll 1$ .



# Application: cage model for gel electrophoresis of DNA



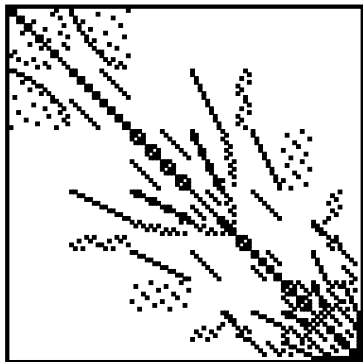
- ▶ 3D cubic lattice models a gel, in which a DNA polymer moves.
- ▶ DNA reptates, i.e., moves like a snake: its kinks and end points move under the influence of an electric field  $E$ .
- ▶ Gel electrophoresis was used in first-generation DNA sequencing to separate DNA fragments by length.
- ▶ Shorter fragments move faster.



A. van Heukelum, G. T. Barkema, R. H. Bisseling, *Journal of Computational Physics* **180** (2002) pp. 313–326. Lecture 4.1 Sequential Sparse Matrix-Vector Multiplication



## Transition matrix for the cage model

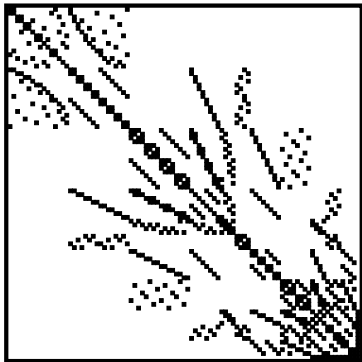


- ▶ Matrix element  $a_{ij}$  is the **probability** that a polymer in state  $j$  moves to a state  $i$ .
- ▶ Hence, the matrix is **stochastic**, i.e.  $0 \leq a_{ij} \leq 1$  and

$$\sum_{i=0}^{n-1} a_{ij} = 1, \text{ for all } j.$$

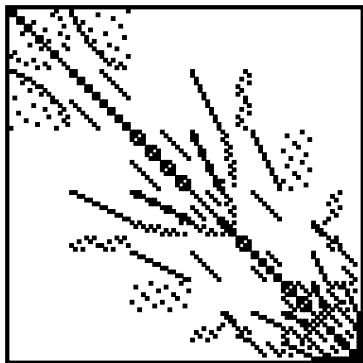


## Transition matrix is sparse



- ▶ Polymer has 6 **monomers** for cage6. We can move only one monomer at a time.
- ▶ Hence, each state has only a **few connected states** and the matrix is sparse.

## Sparsity structure of cage6



- ▶ Each move can be reversed, hence  $a_{ij} \neq 0 \iff a_{ji} \neq 0$ , i.e., the matrix is **structurally symmetric**.
- ▶ A move against the electric field has a different probability than a move with the field. Hence  $a_{ij} \neq a_{ji}$ , so that the matrix is **unsymmetric**.





# Power method

- ▶ Let  $\mathbf{x}$  be the vector of state frequencies: component  $x_i$  represents the relative **frequency** of state  $i$ , with  $0 \leq x_i \leq 1$  and  $\sum_i x_i = 1$ .
- ▶ The **power method** computes  $A\mathbf{x}, A^2\mathbf{x}, A^3\mathbf{x}, \dots$ , until convergence.
- ▶ The final component  $x_i$  represents the frequency of state  $i$  in the **steady-state** situation, where  $A\mathbf{x} = \mathbf{x}$ .
- ▶ Main operation: multiplication of sparse matrix  $A$  and dense vector  $\mathbf{x}$ .



# Sparse matrix–vector multiplication (SpMV)

- ▶ Let  $A$  be a sparse  $n \times n$  matrix and  $\mathbf{v}$  a dense input vector of length  $n$ .
- ▶ We consider the problem of computing the dense output vector  $\mathbf{u}$ ,

$$\mathbf{u} := A\mathbf{v}.$$

- ▶ The components of  $\mathbf{u}$  are

$$u_i = \sum_{j=0}^{n-1} a_{ij}v_j, \quad \text{for } 0 \leq i < n.$$



# Sparse matrix–vector multiplication algorithm

*input:*  $A$  : sparse  $n \times n$  matrix,  
 $\mathbf{v}$  : dense vector of length  $n$ .

*output:*  $\mathbf{u}$  : dense vector of length  $n$ ,  $\mathbf{u} = A\mathbf{v}$ .

**for**  $i := 0$  **to**  $n - 1$  **do**

$u_i := 0$ ;

**for all**  $(i, j) : 0 \leq i, j < n \wedge a_{ij} \neq 0$  **do**

$u_i := u_i + a_{ij}v_j$ ;

- ▶ The sparsity of  $A$  is expressed by the test  $a_{ij} \neq 0$ .
- ▶ Such a test is never executed in practice, and instead a **sparse data structure** is used.

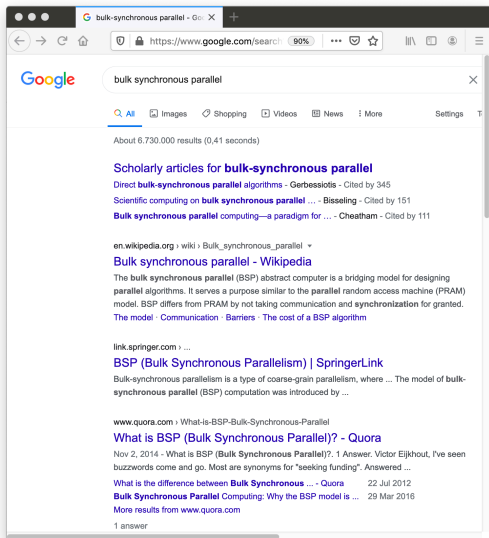


# Application of SpMV: iterative solution methods

- ▶ Sparse matrix–vector multiplication is the **main computation step** in iterative solution methods for linear systems or eigensystems.
- ▶ Iterative methods start with an **initial guess**  $\mathbf{x}^0$  and then successively improve the solution by finding **better approximations**  $\mathbf{x}^k$ ,  $k = 1, 2, \dots$ , until the error is tolerable.
- ▶ Examples:
  - ▶ Linear systems  $A\mathbf{x} = \mathbf{b}$ , solved by the conjugate gradient (CG) method or MINRES, GMRES, QMR, BiCG, Bi-CGSTAB, IDR, IDR( $s$ ), SOR, FOM, ...
  - ▶ Eigensystems  $A\mathbf{x} = \lambda\mathbf{x}$  solved by the Lanczos method, Jacobi–Davidson, ...
- ▶ **One size does not fit all.** Different applications require different iterative methods.



# Web searching: which page ranks first?



The screenshot shows a Google search results page for the query "bulk-synchronous parallel". The search bar at the top contains the text "bulk synchronous parallel". Below the search bar, there are navigation options: "All", "Images", "Shopping", "Videos", "News", and "More". The search results are displayed in a list format. The first result is a link to "en.wikipedia.org" with the title "Bulk synchronous parallel - Wikipedia". The second result is a link to "link.springer.com" with the title "BSP (Bulk Synchronous Parallelism) | SpringerLink". The third result is a link to "www.quora.com" with the title "What is BSP (Bulk Synchronous Parallel)? - Quora". The search results are ranked based on their relevance to the query, with the Wikipedia page being the top result.

bulk-synchronous parallel - Google

https://www.google.com/search?q=bulk+synchronous+parallel

Google

bulk synchronous parallel

Search All Images Shopping Videos News More Settings

About 6.730.000 results (0,41 seconds)

**Scholarly articles for bulk-synchronous parallel**

Direct **bulk-synchronous parallel** algorithms - Gerbessiotis - Cited by 345

Scientific computing on **bulk synchronous parallel** ... - Bisseling - Cited by 151

**Bulk synchronous parallel** computing—a paradigm for ... - Cheatham - Cited by 111

en.wikipedia.org › wiki › Bulk\_synchronous\_parallel

**Bulk synchronous parallel - Wikipedia**

The **bulk synchronous parallel** (BSP) abstract computer is a bridging model for designing **parallel** algorithms. It serves a purpose similar to the **parallel** random access machine (PRAM) model. BSP differs from PRAM by not taking communication and **synchronization** for granted.

The model · Communication · Barriers · The cost of a BSP algorithm

link.springer.com › ...

**BSP (Bulk Synchronous Parallelism) | SpringerLink**

Bulk-synchronous parallelism is a type of coarse-grain parallelism, where ... The model of **bulk-synchronous parallel** (BSP) computation was introduced by ...

www.quora.com › What-is-BSP-Bulk-Synchronous-Parallel

**What is BSP (Bulk Synchronous Parallel)? - Quora**

Nov 2, 2014 - What is BSP (Bulk Synchronous Parallel)? 1 Answer. Victor Eijkhout, I've seen buzzwords come and go. Most are synonyms for "seeking funding". Answered ...

What is the difference between **Bulk Synchronous** ... - Quora 22 Jul 2012

**Bulk Synchronous Parallel** Computing: Why the BSP model is ... 29 Mar 2016

More results from www.quora.com

1 answer



# The weblink matrix $A$

- ▶ Given  $n$  web pages with hyperlinks between them, we can define the sparse  $n \times n$  **weblink matrix**  $A$  by

$$a_{ij} = \begin{cases} 1 & \text{if there is a hyperlink from page } j \text{ to page } i \\ 0 & \text{otherwise.} \end{cases}$$

- ▶ Let  $\mathbf{e} = (1, 1, \dots, 1)^T$ , represent an initial uniform importance (rank) of all web pages. Then

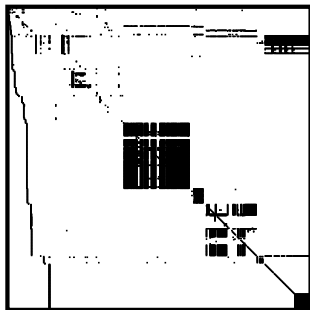
$$(\mathbf{A}\mathbf{e})_i = \sum_j a_{ij}e_j = \sum_j a_{ij}$$

is the **total number of hyperlinks pointing to page  $i$** .

- ▶ The vector  $\mathbf{A}\mathbf{e}$  represents the importance of the pages;  $\mathbf{A}^2\mathbf{e}$  takes the importance of the pointing pages into account as well; and so on.



## Weblink matrix bspww500



- ▶ This matrix with  $n = 500$  and  $nz = 13\,400$  represents 500 web pages and the hyperlinks connecting them.
- ▶ It was obtained by a [breadth-first search](#) in 2017 of the World Wide Web starting at <http://www.bsp-worldwide.org> and using the [web crawler](#) `surfer.m` by Cleve Moler.



# The random surfer model

- ▶ A **random web surfer** chooses each of the  $c_j$  **outgoing hyperlinks** from page  $j$  with equal probability  $\frac{1}{c_j}$ .
- ▶ To incorporate this behaviour, we define the  $n \times n$  diagonal **scaling matrix**  $D$  by

$$d_{jj} = c_j,$$

and multiply the weblink matrix  $A$  from the right by  $D^{-1}$ .

- ▶ This divides each column  $j$  of  $A$  by  $c_j$ .





# The Google matrix

- ▶ Let  $\alpha$  be the probability that a surfer follows an outlink of the current page. Typically  $\alpha = 0.85$ . The surfer **jumps to a random page** with probability  $1 - \alpha$ .
- ▶ The **Google** matrix is defined by

$$G = \alpha AD^{-1} + (1 - \alpha) \frac{1}{n} \mathbf{e}\mathbf{e}^T.$$

- ▶ Note that this definition is under the assumption that all  $c_j > 0$ .
- ▶ The PageRank of a set of web pages is obtained by repeated multiplication by  $G$ , involving sparse matrix–vector multiplication by  $A$  and some vector operations.



S. Brin and L. Page, *Computer Networks and ISDN Systems*, **30**(1–7) (1998) pp. 107–117.



# Vector operation

- ▶ The vector  $\mathbf{e}$  can be viewed as an  $n \times 1$  matrix of all ones and the vector  $\mathbf{e}^T$  as a  $1 \times n$  matrix.
- ▶ The matrix  $\mathbf{e}\mathbf{e}^T$  is an  $n \times n$  matrix with all elements equal to 1.
- ▶ Multiplication of a vector  $\mathbf{x}$  by  $\mathbf{e}\mathbf{e}^T$  is cheap:

$$\mathbf{e}\mathbf{e}^T \mathbf{x} = \mathbf{e}(\mathbf{e}^T \mathbf{x}) = \left( \sum_{i=0}^{n-1} x_i \right) \mathbf{e}.$$



## Escaping from dangling nodes

- ▶ If  $c_j = 0$ , column  $j$  must be empty and page  $j$  a **dangling node**; it could be a PDF file or an image file.
- ▶ To avoid division by zero, the diagonal element is then **redefined** as  $d_{jj} = 1$ .
- ▶ To make the Google matrix stochastic (with all column sums equal to 1), we must add to  $G$  an **extra term**

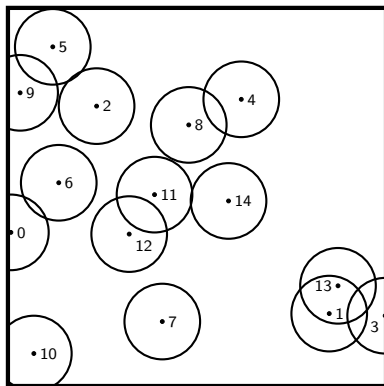
$$\alpha \frac{1}{n} \mathbf{e} \hat{\mathbf{e}}^T.$$

- ▶ Here, the vector  $\hat{\mathbf{e}}$  is defined by

$$\hat{e}_j = \begin{cases} 1 & \text{if } c_j = 0 \\ 0 & \text{otherwise.} \end{cases}$$



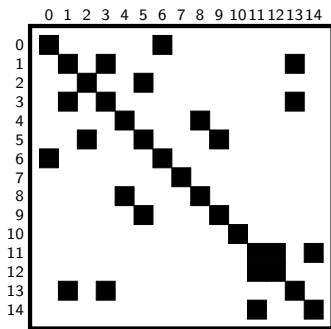
## Insight into other applications



- ▶ A **molecular dynamics** domain of size  $1.0 \times 1.0$  with 15 particles.
- ▶ The **cut-off radius** for interaction between particles is  $r_c = 0.2$ . The circles shown have radius  $r_c/2 = 0.1$ .
- ▶ Particles interact if their circles intersect.



# Force matrix $F$ represents particle interactions



- ▶ Shown is the matrix  $F + I$ , where  $F$  is the sparse  $15 \times 15$  **force matrix** corresponding to the particle interactions.
- ▶ If particles  $i$  and  $j$  interact, nonzeros  $f_{ij}$  and  $f_{ji}$  appear in  $F$ .
- ▶ Row  $i$  of  $F + I$  expresses the **information needed** to compute the next position of particle  $i$  in a molecular dynamics simulation: forces  $f_{ij} \neq 0$  and the current position and velocity of  $i$ .



# Summary

- ▶ Sparse matrices are the **rule, rather than the exception**. In many applications, variables are connected to only a few others, leading to sparse matrices.
- ▶ Sparse matrices occur in many application areas:
  - ▶ transition matrices in Markov models;
  - ▶ finite-element matrices in engineering;
  - ▶ linear programming matrices in optimization;
  - ▶ weblink matrices in Google PageRank computation.
- ▶ Sparse matrix–vector multiplication is important for **iterative solvers** and it is the workhorse of the PageRank computation.
- ▶ It can also capture **other applications** such as molecular dynamics.
- ▶ The sequential computation is simple, but its parallelization is a **challenge**.

