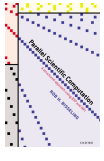
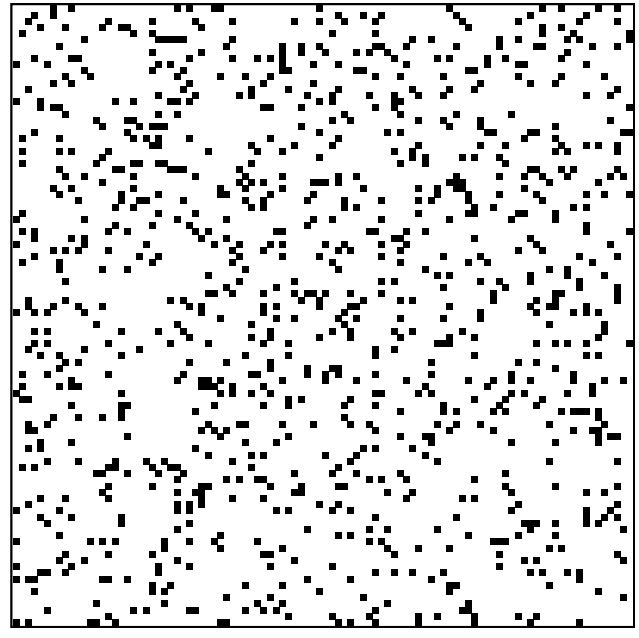
# *Random Sparse Matrices*
# *(PSC §4.7)*

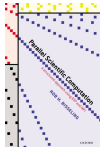


# Random sparse matrix `random100`



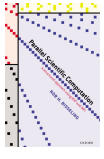
$n = 100$, $nz = 1000$, $c = 10$, $d = 0.1$.
Interactively generated at the Matrix Market Deli (Boisvert *et al.* 1997),

`http://math.nist.gov/MatrixMarket/deli/Random/`

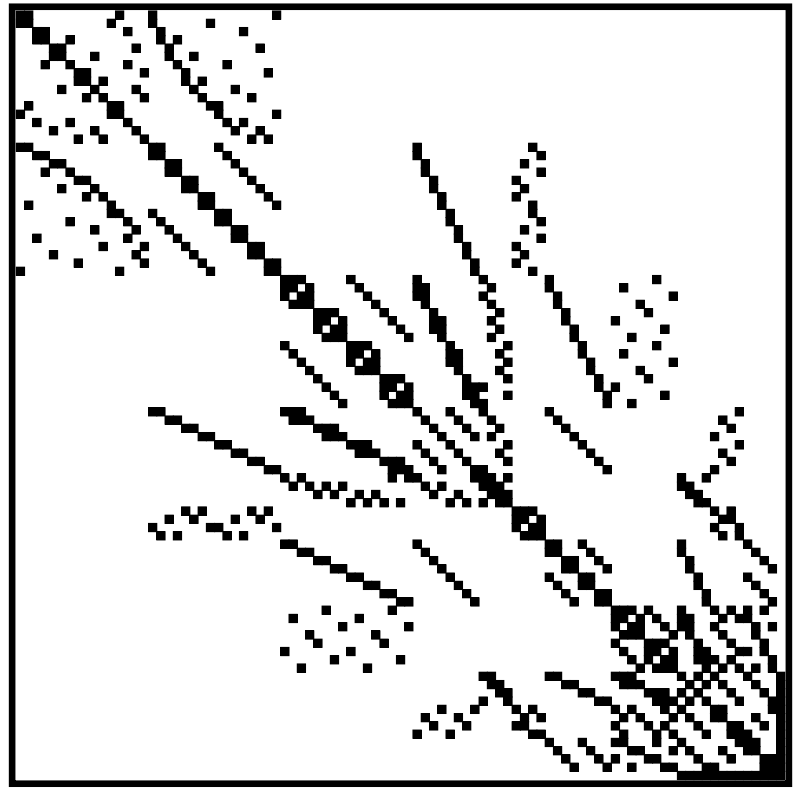


# Random sparse matrix

- A random sparse matrix $A$ can be obtained by determining, randomly and independently, for each element $a_{ij}$ whether it is 0 or not.
- If the probability of creating a nonzero is $d$, the matrix has:
  - an expected density $d(A) = d$;
  - an expected number of nonzeros $nz(A) = dn^2$.
- Random sparse matrices have a very special property: every subset of the matrix elements, chosen independently from the sparsity pattern, has an expected fraction $d$ of nonzeros.
- This property provides a powerful tool for analysing algorithms involving random sparse matrices.

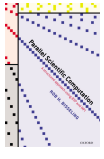


# *Not a random sparse matrix*



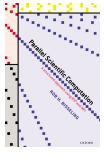
Matrix cage6 from DNA electrophoresis studies.

- Some structure immediately visible.
- Don’t use the term ‘random sparse matrix’ for such a matrix or a sparse matrix without any visible structure.



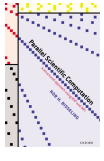
# Parallel sparse matrix–vector multiplication

- Construct a random sparse matrix $A$ by drawing for each index pair $(i, j)$ a random number $r_{ij} \in [0, 1]$, doing this independently and uniformly (with each outcome equally likely), creating a nonzero $a_{ij}$ if $r_{ij} < d$.
- Distribute $A$ over $p$ processors in a manner that is independent of the sparsity pattern by assigning an equal number of elements (whether 0 or not) to each processor.
- Examples are the square block distribution, square cyclic distribution, and the cyclic row distribution.



# Computational load balance

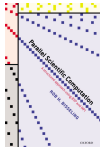
- The load balance can be estimated by using probability theory.
- The problem is to determine the expected maximum, taken over all processors, of the local number of nonzeros.
- We cannot solve this problem exactly, but we can obtain a useful bound on the probability of the maximum exceeding a certain value.
- Bound is obtained by applying a theorem of Chernoff, often used in the analysis of randomised algorithms.



# Theorem 4.11 (Chernoff 1952)

- Let $0 < d < 1$.
- Let $X_0, X_1, \ldots, X_{m-1}$ be independent Bernoulli trials with outcome 0 or 1, such that $\mathbf{Pr}[X_k = 1] = d$, for $0 \leq k < m$.
- Let $X = \sum_{k=0}^{m-1} X_k$ and $\mu = md$.
- Then for every $\epsilon > 0$,

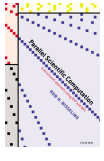
$$\mathbf{Pr}[X > (1+\epsilon)\mu] < \left( \frac{\mathrm{e}^{\epsilon}}{(1+\epsilon)^{1+\epsilon}} \right)^{\mu}.$$



# Application of Chernoff Theorem

$$\mathbf{Pr}[X > (1+\epsilon)\mu] < \left(\frac{\mathrm{e}^{\epsilon}}{(1+\epsilon)^{1+\epsilon}}\right)^{\mu}.$$

- If we flip a biased coin which produces heads with probability $d$, the bound tells us how small the probability is of getting $\epsilon\mu$ more heads than the expected average $\mu$.
- Bound for $\epsilon = 1$ tells us that the probability of getting more than twice the expected number of heads is less than $(\mathrm{e}/4)^{\mu} \approx (0.68)^{md}$.



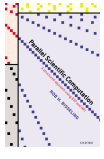
# Application to random sparse matrix

- Every processor has $m = \frac{n^2}{p}$ elements, each being nonzero with probability $d$.
- Expected number of nonzeros per processor is $\mu = \frac{dn^2}{p}$.
- Let $E_s$ be the event that processor $P(s)$ has more than $(1+\epsilon)\mu$ nonzeros. Let $E = \cup_{s=0}^{p-1} E_s$.
- We have

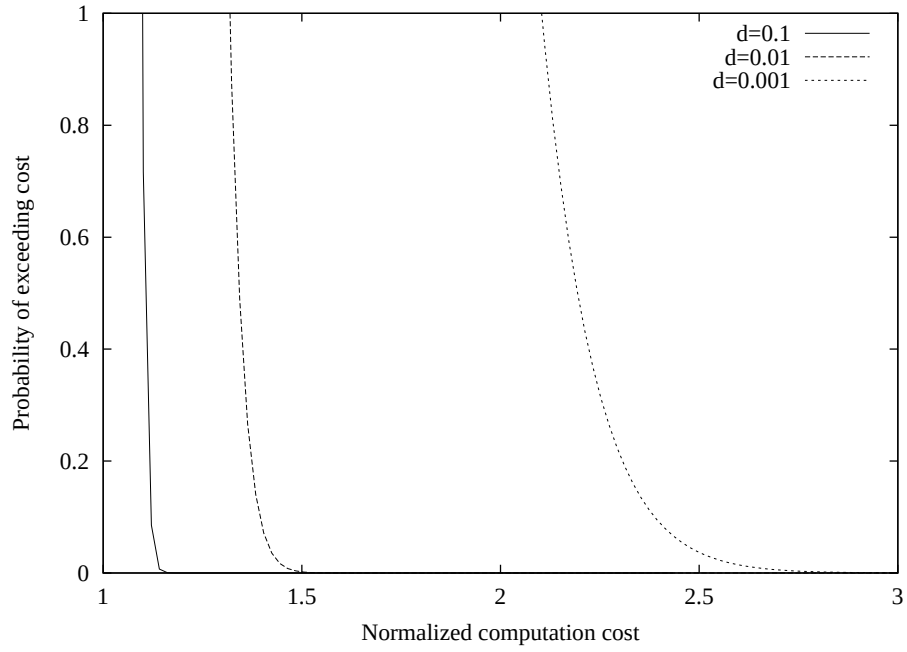
$$\mathbf{Pr}[E] \leq \sum_{s=0}^{p-1} \mathbf{Pr}[E_s] = p\mathbf{Pr}[E_0],$$

so that the cost of superstep (1) satisfies

$$\mathbf{Pr}\left[T_{(1)} > \frac{2(1+\epsilon)dn^2}{p}\right] < p\left(\frac{\mathrm{e}^{\epsilon}}{(1+\epsilon)^{1+\epsilon}}\right)^{\frac{dn^2}{p}}.$$

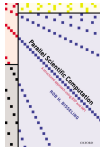


# Probability of exceeding normalised cost

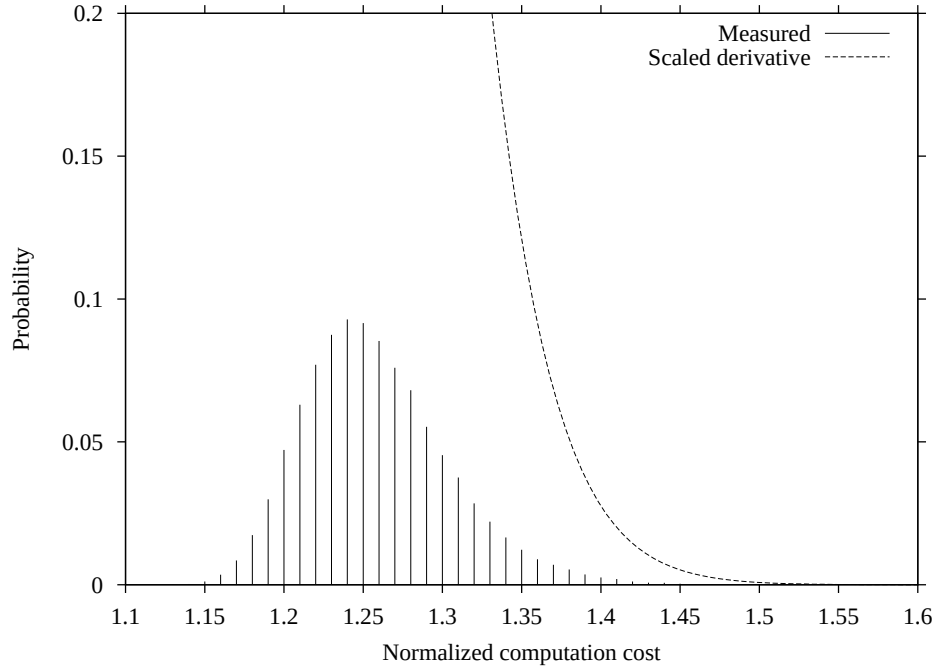



Chernoff probability $F(\epsilon)$ of exceeding normalised cost $1+\epsilon$ for random sparse matrix of size $n = 1000$ and density $d$ distributed over $p = 100$ processors.
Average normalised cost obtained by simulation:

1.076 for $d = 0.1$; 1.258 for $d = 0.01$; 1.876 for $d = 0.001$.

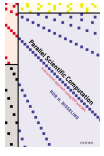


# Probability distribution



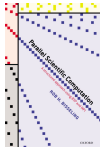

Probability distribution obtained by simulation for random sparse matrix of size $n = 1000$ and density $d = 0.01$ distributed over $p = 100$ processors.

- Local nonzero count 124 (cost = 1.24) occurs most.
- Derivative is $(1 - F(\epsilon))'$, probability density function corresponding to the Chernoff bound. Pessimistic!

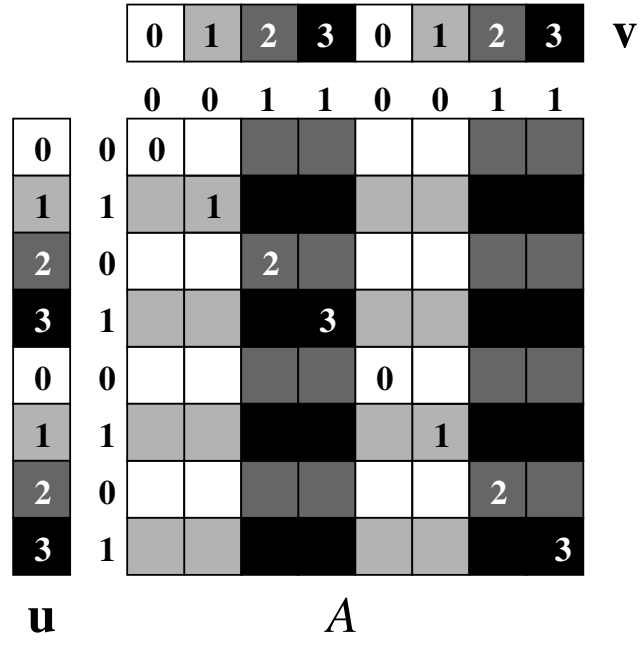


# *Communication cost for random sparse matrix*

- The communication volume for a dense matrix is an upper bound on the volume for a sparse matrix distributed by the same fixed, pattern-independent scheme.
- For a random sparse matrix with a high density, the communication obligations will be the same as for a dense matrix.
- Therefore, we try to find a good fixed distribution scheme for random sparse matrices by applying methods from the dense case.

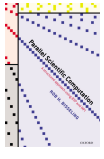


# Square Cartesian distribution for dense matrix




$n = 8$, $p = 4$, $M = N = 2$.

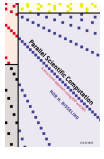
Square Cartesian distribution based on a cyclic distribution of the matrix diagonal.



# Superstep (0): fanout

- Vector component $v_j$ is needed only in $P(*, \phi_1(j))$.
- $P(s, \phi_1(j))$ does not need $v_j$ if all $\frac{n}{\sqrt{p}}$ elements in the local part of matrix column $j$ are zero; this has probability $(1-d)^{n/\sqrt{p}}$.
- Probability that $P(s, \phi_1(j))$ needs $v_j$ is $1-(1-d)^{n/\sqrt{p}}$.
- Since $\sqrt{p}-1$ processors each have to receive $v_j$ with this probability, the expected number of receives for component $v_j$ is $(\sqrt{p}-1)(1-(1-d)^{n/\sqrt{p}})$.
- Expected volume is $n(\sqrt{p}-1)(1-(1-d)^{n/\sqrt{p}})$.
- Ignoring communication imbalance, we divide by $p$, giving

$$T_{(0)} = \left(\frac{1}{\sqrt{p}} - \frac{1}{p}\right)(1-(1-d)^{n/\sqrt{p}})ng.$$

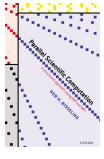


# Total communication cost

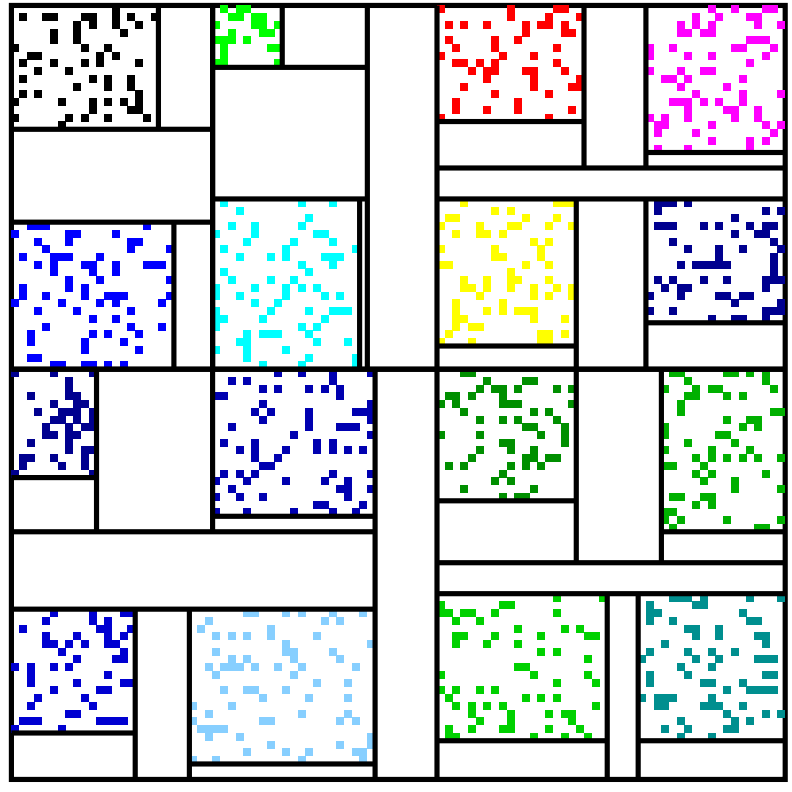
- Cost of fanin is same as for fanout.
- For $n = 1000$ and $p = 100$, the matrix with highest density $d = 0.1$ has an expected communication cost of $179.995g$, close to the cost of $180g$ for a dense matrix.
- The corresponding expected normalised communication cost is

$$\frac{T_{(0)} + T_{(2)}}{2dn^2/p} \approx 0.09g.$$

- We need a parallel computer with $g \leq 11$ to run our algorithm with more than 50% efficiency
- For $n = 1000$ and $p = 100$, the matrix with lowest density $d = 0.001$ has an expected normalised communication cost of $0.86g$.

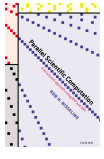


# Tailor the distribution to the matrix



Local view of `random100` (with $n = 100$, $nz = 1000$, $d = 0.1$), distributed by the Mondriaan program over $p = 16$ processors. Shown is the submatrix $I_s \times J_s$ for $0 \leq s < 16$.

- Allowed imbalance $\epsilon = 20\%$; $\epsilon' = 18.4\%$ achieved.
  Max nonzeros per proc 74. Avg 62.5. Min 25. $V = 367$.



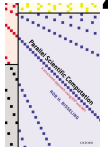
# Comparing Cartesian and Mondriaan distribution

| $p$ | $\epsilon$ (in %) | $V$ (Cartesian) | $V$ (Mondriaan) |
|---|---|---|---|
| 2 | 0.8 | 993 | 814 |
| 4 | 2.1 | 1987 | 1565 |
| 8 | 4.0 | 3750 | 2585 |
| 16 | 7.1 | 5514 | 3482 |
| 32 | 11.8 | 7764 | 4388 |

Random sparse matrix of size $n = 1000$ and density $d = 0.01$ distributed over $p$ processors, by:

- pattern-independent Cartesian distribution;
- pattern-dependent distribution produced by the Mondriaan package with $\epsilon =$ expected value for the Cartesian distribution.

45% less communication volume for Mondriaan ($p = 32$).



# Summary

- Distributing a random sparse matrix independently of its sparsity pattern spreads the computation well.
- We can quantify this by using the Chernoff bound

$$\mathbf{Pr}[X > (1+\epsilon)\mu] < \left(\frac{\mathrm{e}^{\epsilon}}{(1+\epsilon)^{1+\epsilon}}\right)^{\mu}.$$

- For the communication, we can use a pattern-independent square Cartesian distribution which distributes the matrix diagonal and the vectors cyclically over the processors.
- The distribution can be improved by tailoring it to the sparsity pattern e.g. by using Mondriaan.
- Parallel multiplication of a random sparse matrix and a vector remains a difficult problem.