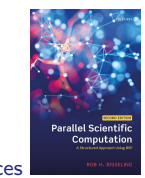
Random Sparse Matrices

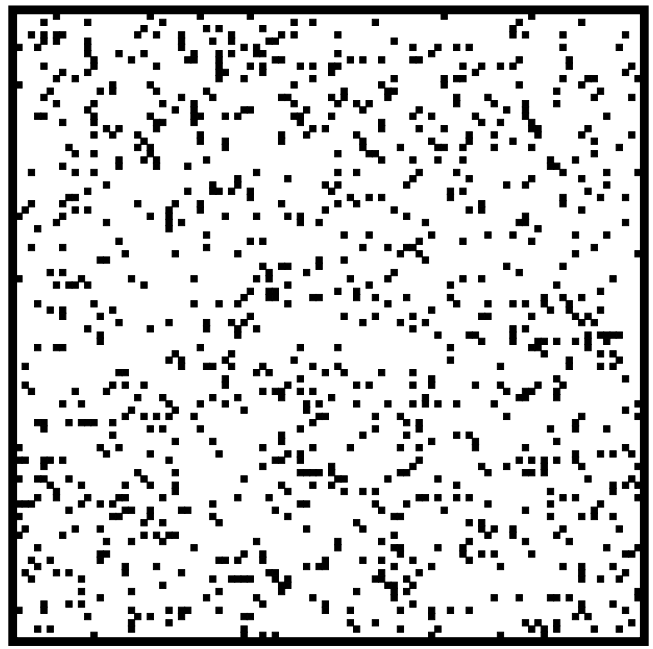
Section 4.8 of Parallel Scientific Computation, 2nd edition

Rob H. Bisseling

Utrecht University

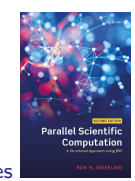


Sparse matrix `random100`



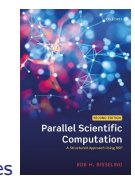
- Random sparse matrix with $n = 100$, $nz = 982$, $d = 0.0982$, created by using the random number generator `Ran` from Numerical Recipes.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, Third edition, 2007, Cambridge University Press.

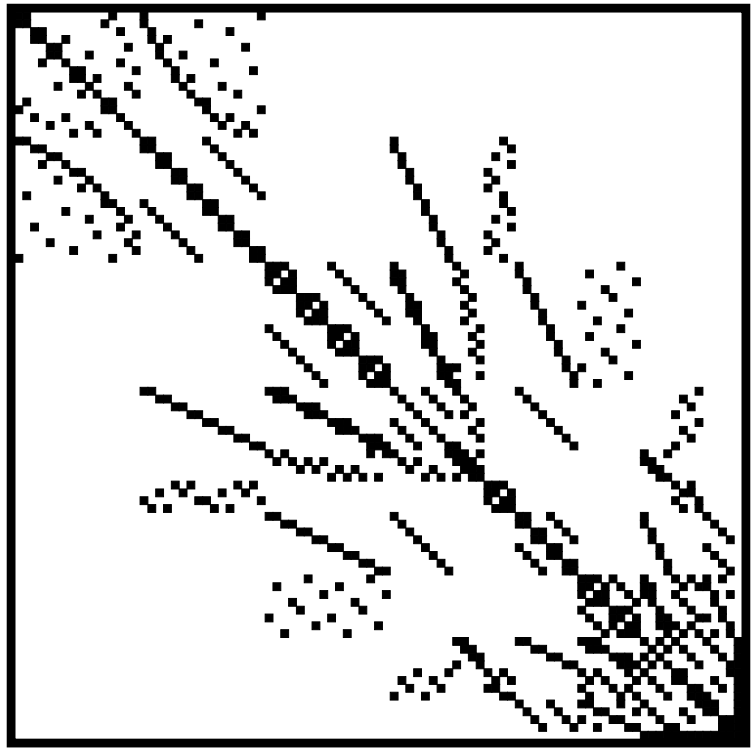


Generating a random sparse matrix

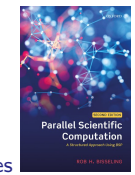
- A random sparse matrix A can be generated by determining, randomly and independently, for each element a_{ij} whether it is 0 or not.
- If the probability of creating a nonzero is d, the matrix has:
  - an expected density $d(A) = d$;
  - an expected number of nonzeros $nz(A) = dn^2$.
- Random sparse 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.



A structured, nonrandom sparse matrix: cage6

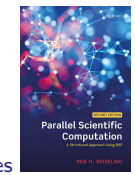


- Urban Dictionary: a 'random person' often means an unknown, unexpected, or unfamiliar person.
- Google can help find a (truly?) random person.
- However, don't use the term 'random sparse matrix' for a sparse matrix with a structure that is unfamiliar or not immediately visible.



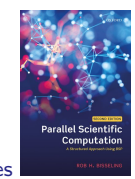
Parallel sparse matrix–vector multiplication

- Generate a random sparse matrix A by drawing for each index pair (i, j) a real 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 independent of the sparsity pattern by assigning an equal number of elements (whether 0 or not) to each processor.
- Examples are:
  - square block distribution;
  - square cyclic distribution;
  - 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.
- The bound is obtained by applying the Chernoff Theorem, often used in the analysis of randomized algorithms.

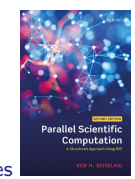


Theorem 4.10 (Chernoff)

- 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 $\epsilon > 0$,

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

H. Chernoff, *Annals of Mathematical Statistics*, **23**(4) (1952), pp.493–507.

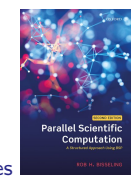


Probability of $X > 2\mu$

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

- The bound for $\epsilon = 1$ tells us that the probability of getting more than twice the expected number μ is

$$\mathbf{Pr}[X > 2\mu] < \left(\frac{\mathrm{e}}{4} \right)^{\mu} \approx (0.68)^{md}.$$



Application to a random sparse matrix

- The expected number of nonzeros per processor is

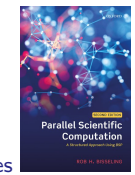
$$\mu = \frac{dn^2}{p}.$$

- Let E_s be the event that processor $P(s)$ has more than $(1+\epsilon)\mu$ nonzeros and let

$$E = \bigcup_{s=0}^{p-1} E_s.$$

- Let $q = \mathbf{Pr}[E_s]$, which is the same for all s.
- By looking at the probability that no event E_s occurs in any processor, we obtain the equality

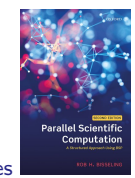
$$\mathbf{Pr}[E] = 1 - (1-q)^p.$$



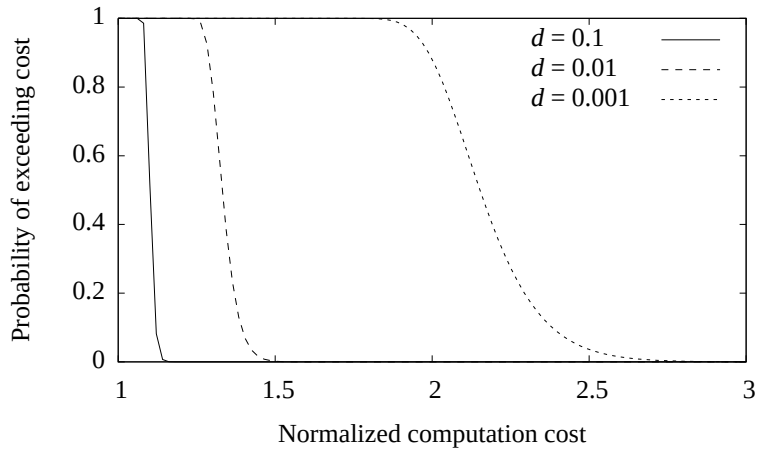
Cost of the local sparse matrix–vector multiplication

- The cost $T_{(1)}$ of superstep (1) satisfies

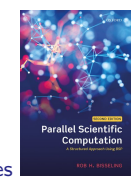
$$\begin{aligned}\mathbf{Pr}\left[T_{(1)} > \frac{2(1+\epsilon)dn^2}{p}\right] &= 1-(1-q)^p \\ &< 1-\left(1-\left(\frac{\mathrm{e}^{\epsilon}}{(1+\epsilon)^{1+\epsilon}}\right)^{\frac{dn^2}{p}}\right)^p \\ &= F(\epsilon).\end{aligned}$$



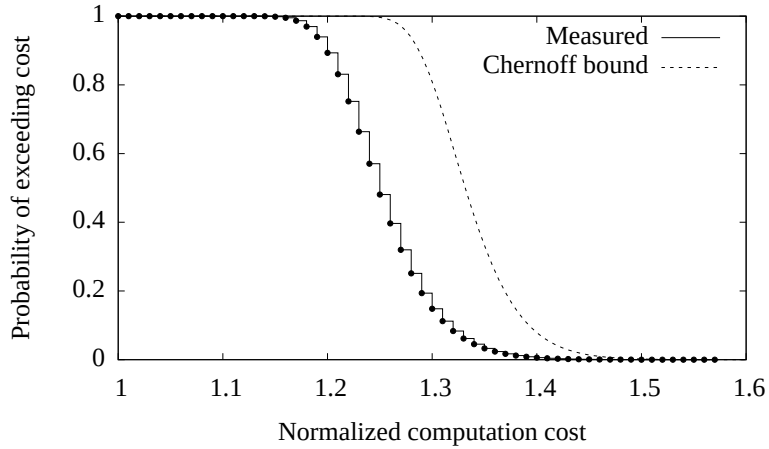
Bound on probability of exceeding the normalized cost



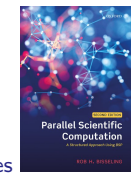
- Shown is the Chernoff probability $F(\epsilon)$ of exceeding the normalized cost $1+\epsilon$ for a random sparse matrix of size $n = 1000$ and density d distributed over $p = 100$ processors.
- The average normalized cost obtained by simulation for 10 000 matrices is:
  - 1.076 for $d = 0.1$;
  - 1.258 for $d = 0.01$;
  - 1.876 for $d = 0.001$.



Measured probability of exceeding the normalized cost

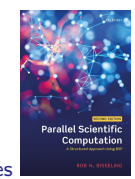


- Shown is the measured probability of exceeding the normalized cost $1 + \epsilon$ for a random sparse matrix with $n = 1000$, $d = 0.01$, $p = 100$, based on 100 000 matrices.
- For comparison, also the corresponding (pessimistic) Chernoff bound is given.
- A maximum local nonzero count 124 (i.e., cost $= 1.24$) occurs most often, with a frequency of 9.3%.
- The probability that one of the processors has more than 124 nonzeros is 57.1%.

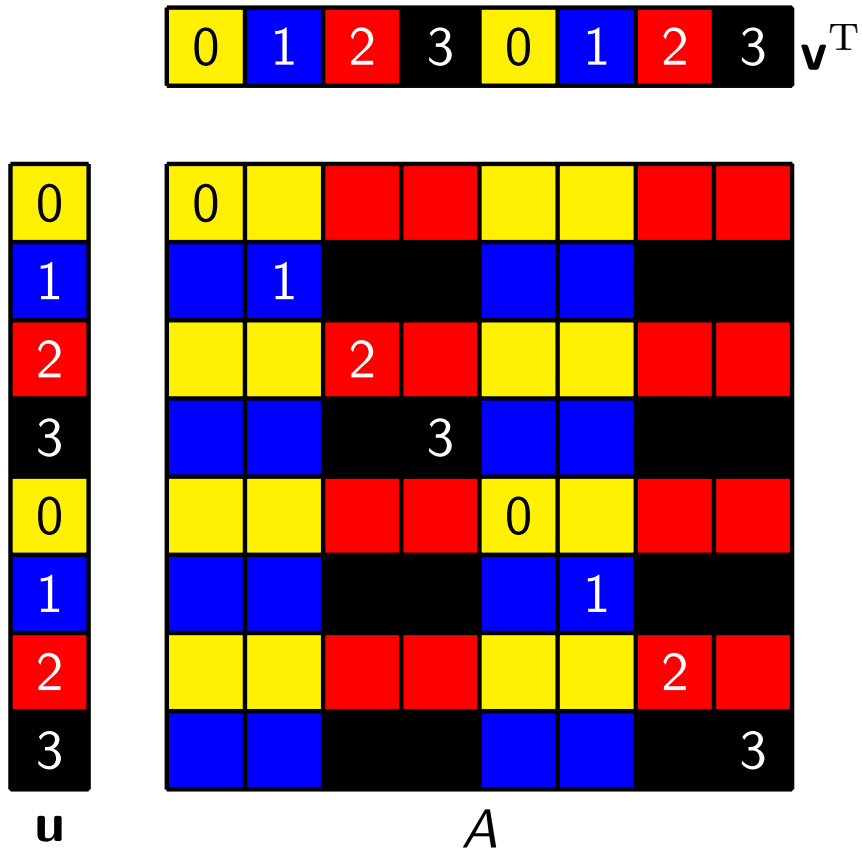


Communication cost for a random sparse matrix

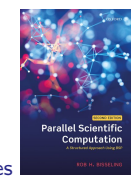
- The communication volume for a dense matrix is an upper bound on the volume for a sparse matrix distributed by the same fixed, pattern-independent scheme.
- The communication obligations for a random sparse matrix with a high density will almost be the same as for a dense matrix.
- Therefore, we can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



Square Cartesian distribution for a dense matrix



- $n = 8,\ p = 4.$
- Square 2×2 Cartesian distribution based on a cyclic distribution of the matrix diagonal.



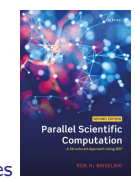
Superstep (0): fanout

- 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 $(1-d)^{n/\sqrt{p}}$.
- The probability that $P(s, \phi_1(j))$ needs v_j is $1-(1-d)^{n/\sqrt{p}}$.
- Since $\sqrt{p}-1$ off-diagonal processors each have to receive v_j with this probability, the expected number of receives for component v_j is $(\sqrt{p}-1)(1-(1-d)^{n/\sqrt{p}})$.
- Hence, the expected communication volume for the fanout is

$$n(\sqrt{p}-1)(1-(1-d)^{n/\sqrt{p}}).$$

- Ignoring communication imbalance, we divide by p, giving

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

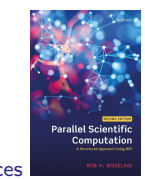


Total communication cost

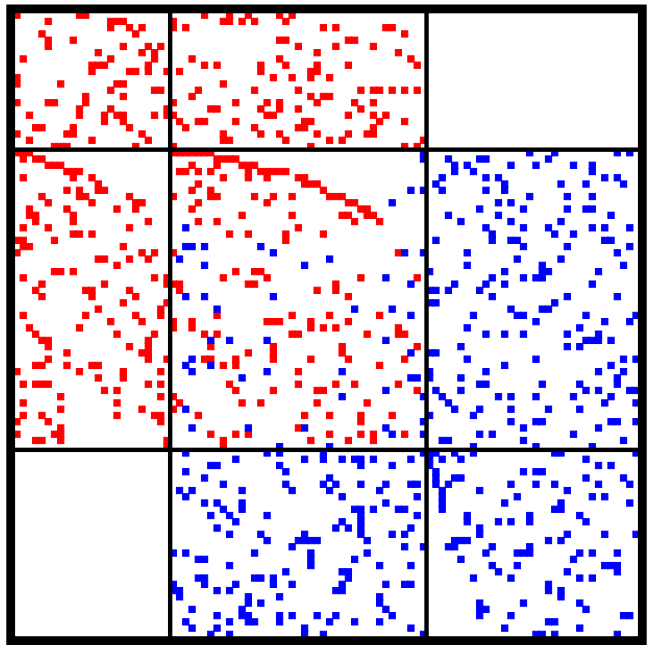
- Cost of the fanin is the same as for the fanout.
- For $n = 1000$ and $p = 100$, the matrix with highest density $d = 0.1$ has an expected communication cost of $179.995g$, close to the cost of $180g$ for a dense matrix.
- The corresponding expected normalized communication cost is

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

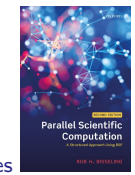
- We need a parallel computer with $g \leq 11$ to run our algorithm with more than 50% efficiency.
- For $n = 1000$ and $p = 100$, the matrix with lowest density $d = 0.001$ has an expected normalized communication cost of $0.86g$.



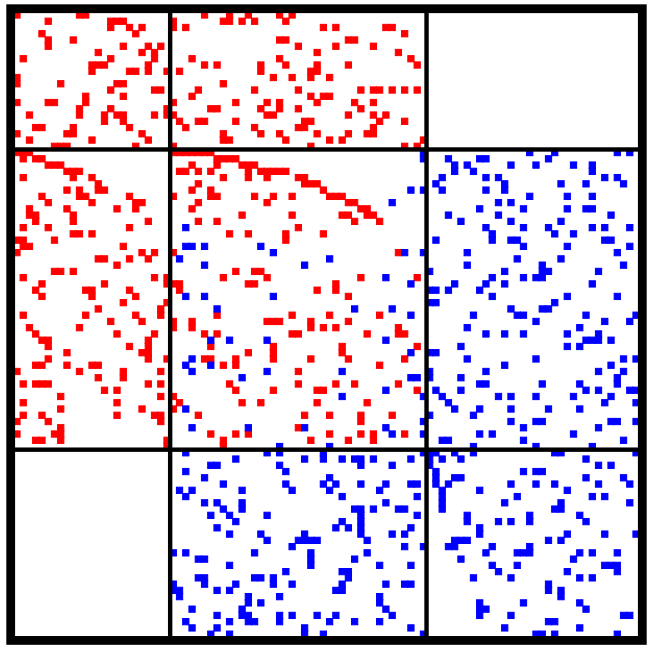
Tailor the distribution to the matrix



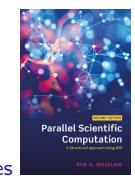
- Global permuted view of the sparse matrix `random100` with $n = 100$ and $nz = 982$, distributed for $p = 2$ and $\epsilon = 0.03$ by Mondriaan v4.2 with the medium-grain method.
- The matrix is shown in Separated Block Diagonal (SBD) form with the 48 cut rows and 41 cut columns in the middle.
- The BSP cost is $982 + 45g + 4l$.



Separated Block Diagonal (SBD) form



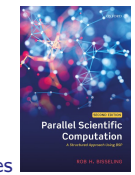
- The SBD form is useful for visualizing the communication requirements of a parallel SpMV.
- It can also be used to speed up a sequential SpMV, by keeping vector components v_j longer in cache.
- The SpMV can start and end in cache, with a gradual transition in between.



Communication volume for Cartesian vs. Mondriaan

p	ϵ (in %)	ϵ' (in %)	V (Cartesian)	V (Mondriaan)
2	0.8	0.005	993	862
4	2.1	0.015	1 987	1 765
8	4.0	0.048	3 750	2 696
16	7.1	4.318	5 514	3 611
32	11.8	10.874	7 764	4 461

- Random sparse matrix of size $n = 1000$ and density $d = 0.01$ distributed over p processors by:
  - pattern-independent Cartesian distribution, with an expected imbalance ϵ;
  - pattern-dependent distribution produced by the Mondriaan package with the allowed imbalance set to ϵ; the value actually achieved by Mondriaan is ϵ'.
- Result for $p = 32$: using Mondriaan reduces the communication by 43%. (But it is still a lot!)



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 the Mondriaan partitioner.
- Parallel multiplication of a random sparse matrix and a vector remains a difficult problem, because there is relatively much communication.