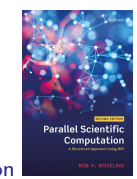
Cartesian Matrix Distribution

Section 4.4 of Parallel Scientific Computation, 2nd edition

Rob H. Bisseling

Utrecht University



Identifying 1D and 2D processor numbering

- Natural column-wise identification for $p = MN$ processors:

$$P(s,t) \equiv P(s + tM), \quad \text{for } 0 \leq s < M \quad \text{and} \quad 0 \leq t < N.$$

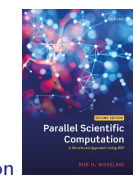
- This can also be written as

$$P(s) \equiv P(s \bmod M, s \operatorname{div} M), \quad \text{for } 0 \leq s < p.$$

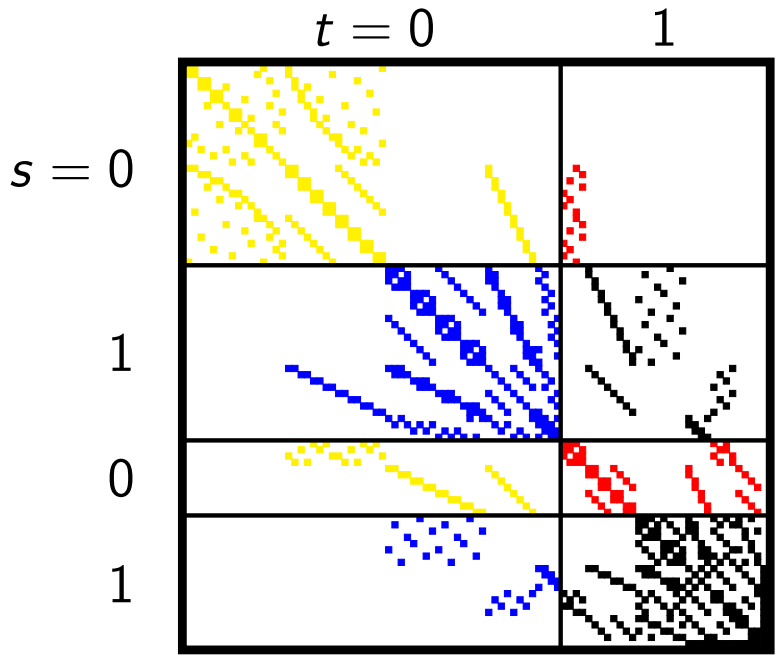
- For a Cartesian distribution (ϕ_0, ϕ_1), we map nonzeros a_{ij} to processors $P(\phi(i,j))$ by

$$\phi(i,j) = \phi_0(i) + \phi_1(j)M, \quad \text{for } 0 \leq i,j < n \quad \text{and} \quad a_{ij} \neq 0.$$

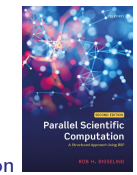
- We use 1D or 2D numbering, whichever is most convenient in the context.



A Cartesian distribution of cage6



- $n = 93$, $nz = 785$, $p = 4$, $M = N = 2$.
- The processor row of a matrix element a_{ij} is $s = \phi_0(i)$; the processor column is $t = \phi_1(j)$.
- The matrix diagonal is assigned in blocks to the processors: $P(0) \equiv P(0,0)$, $P(2) \equiv P(0,1)$, $P(1) \equiv P(1,0)$, $P(3) \equiv P(1,1)$.



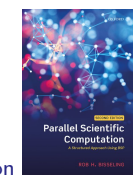
Advantages of a Cartesian distribution for a sparse matrix

Advantages:

- Row-wise operations require communication only within processor rows, and similar for column-wise operations.
- For an $M \times N$ Cartesian distribution, v_j has to be sent to at most M processors and u_i is computed using contributions received from at most N processors.
- Simplicity: Cartesian distributions partition the matrix orthogonally into rectangular submatrices. Non-Cartesian distributions create arbitrarily-shaped matrix parts.

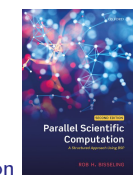
Disadvantage:

- Less general, so a Cartesian distribution may not offer the optimal solution.



Matching the matrix and vector distributions

- In a Cartesian distribution, vector component v_j is needed only by processors that possess an $a_{ij} \neq 0$, and these processors are contained in processor column $P(*, \phi_1(j))$.
- Assigning vector component v_j to one of the processors in $P(*, \phi_1(j))$ implies that v_j has to be sent to at most $M - 1$ processors, instead of M.
- If we are lucky (or clever), we may even avoid communication of v_j altogether.
- If v_j were assigned to a different processor column, it would always have to be communicated.
- Assigning u_i to a processor in processor row $P(\phi_0(i), *)$ reduces the number of contributions sent for u_i to at most $N - 1$.



A trivial but powerful theorem

Theorem 4.4 Let A be a sparse $n \times n$ matrix and $\mathbf{u}, \mathbf{v}$ vectors of length n. Assume that:

(i) the distribution of A is Cartesian, $\mathrm{distr}(A) = (\phi_0, \phi_1)$;

(ii) the distribution of $\mathbf{u}$ is such that u_i resides in $P(\phi_0(i), *)$;

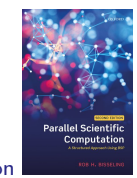
(iii) the distribution of $\mathbf{v}$ is such that v_j resides in $P(*, \phi_1(j))$.

Then: if u_i and v_j are assigned to the same processor, a_{ij} is also assigned to that processor and does not cause communication.

Proof.
Component u_i is assigned to $P(\phi_0(i), t)$.
Component v_j is assigned to $P(s, \phi_1(j))$.
Since this is the same processor, we have

$$(s, t) = (\phi_0(i), \phi_1(j)),$$

so that this processor also owns a_{ij}. □



Special case $\mathrm{distr}(\mathbf{u}) = \mathrm{distr}(\mathbf{v})$

Corollary of Theorem 4.4. The conditions

(i) the distribution of A is Cartesian, $\mathrm{distr}(A) = (\phi_0, \phi_1)$;

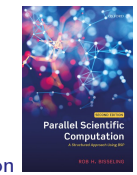
(ii) the distribution of $\mathbf{u}$ is such that u_i resides in $P(\phi_0(i), *)$;

(iii) the distribution of $\mathbf{v}$ is such that v_j resides in $P(*, \phi_1(j))$;

(iv) $\mathrm{distr}(\mathbf{u}) = \mathrm{distr}(\mathbf{v})$;

imply that u_i and v_i are assigned to $P(\phi_0(i), \phi_1(i))$, which is the owner of the diagonal element a_{ii}.

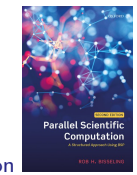
- The choice of a Cartesian matrix distribution completely determines the vector distribution.
- The choice of a vector distribution together with values for M, N completely determines the Cartesian matrix distribution.



Example: 1D Laplacian matrix

$$A = \begin{bmatrix} -2 & 1 & & & & & & \\ 1 & -2 & 1 & & & & & \\ & 1 & -2 & 1 & & & & \\ & & & \ddots & & & & \\ & & & 1 & -2 & 1 & & \\ & & & & 1 & -2 & 1 & \\ & & & & & 1 & -2 & \end{bmatrix}.$$

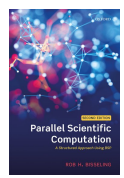
- This tridiagonal matrix represents a Laplacian operator on a 1D grid of n points.
- $a_{ij} \neq 0$ if and only if $i - j = 0, \pm 1$.



Vector distribution for tridiagonal matrix

- $a_{ij} \neq 0$ if and only if $i - j = 0, \pm 1$.
- Assume that we require $\mathrm{distr}(\mathbf{u}) = \mathrm{distr}(\mathbf{v})$. Theorem 4.4 says that it is best to assign u_i and v_j (and hence u_j) to the same processor if $i = j \pm 1$.
- Therefore, a suitable vector distribution over p processors is the block distribution,

$$u_i \longmapsto P\left(i \text{ div } \left\lceil \frac{n}{p} \right\rceil\right), \text{ for } 0 \leq i < n.$$

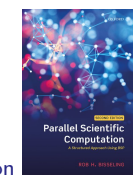


Example: 12×12 1D Laplacian matrix

Distribution matrix for $n = 12$ and $M = N = 2$:

$$\text{distr}(A) = \left[\begin{array}{cccccc|cccccc}
0 & 0 & & & & & & & & & & \\
0 & 0 & 0 & & & & & & & & & \\
 & 0 & 0 & 0 & & & & & & & & \\
\hline
 & & 1 & 1 & 1 & & & & & & & \\
 & & & 1 & 1 & 1 & & & & & & \\
 & & & & 1 & 1 & 3 & & & & & \\
\hline
 & & & & & 0 & 2 & 2 & & & & \\
 & & & & & & 2 & 2 & 2 & & & \\
 & & & & & & & 2 & 2 & 2 & & \\
\hline
 & & & & & & & & 3 & 3 & 3 & \\
 & & & & & & & & & 3 & 3 & 3 \\
 & & & & & & & & & & 3 & 3
\end{array}\right].$$

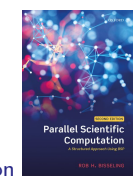
Position (i, j) gives the 1D identity of the processor that owns matrix element a_{ij}.



Construction of the distribution matrix

$\mathrm{distr}(A)$ is constructed by:

- distributing the vectors by the 1D block distribution;
- distributing the matrix diagonal in the same way as the vectors;
- translating the 1D processor numbers into 2D numbers by $P(0) \equiv P(0,0)$, $P(2) \equiv P(0,1)$, $P(1) \equiv P(1,0)$, $P(3) \equiv P(1,1)$;
- determining the owners of the off-diagonal nonzeros:
  - a_{56} is in the same processor row as a_{55}, owned by $P(1) = P(1,0)$;
  - a_{56} is in the same processor column as a_{66}, owned by $P(2) = P(0,1)$;
  - thus, a_{56} is owned by $P(1,1) = P(3)$.



Cost analysis for a Cartesian distribution

- Assume that we have a good spread of
  - nonzeros and vector components over processors;
  - matrix rows over processor rows;
  - matrix columns over processor columns.
- Then the costs of the supersteps are

$$T_{(0)} = (M-1)\frac{ng}{p} + l,$$

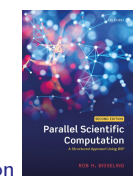
$$T_{(1)} = \frac{2cn}{p} + l,$$

$$T_{(2)} = (N-1)\frac{ng}{p} + l,$$

$$T_{(3)} = \frac{Nn}{p} + l.$$

- The total BSP cost is then bounded by

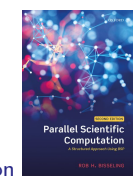
$$T_{\mathrm{MV},\ M\times N} \leq \frac{2cn}{p} + \frac{n}{M} + \frac{M+N-2}{p}ng + 4l.$$



Efficient computation for $M = N = \sqrt{p}$

$$T_{\mathrm{MV},\ \sqrt{p}\times\sqrt{p}} \leq \frac{2cn}{p} + \frac{n}{\sqrt{p}} + 2\left(\frac{1}{\sqrt{p}} - \frac{1}{p}\right) ng + 4l.$$

- The computation is considered efficient if $\frac{2cn}{p} > \frac{2ng}{\sqrt{p}}$, i.e., $c > \sqrt{p}g$.
- This is an improvement of a factor $\sqrt{p}$ compared to the previous general efficiency criterion.

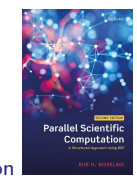


Dense matrices

- Dense matrices are the limit of sparse matrices for $c \to n$.
- Analysing the dense case is easier and it can give us insight into the sparse case as well.
- Substituting $c = n$ in the previous cost formula gives

$$T_{\text{MV, dense}} \leq \frac{2n^2}{p} + \frac{n}{\sqrt{p}} + 2\left(\frac{1}{\sqrt{p}} - \frac{1}{p}\right) ng + 4l.$$

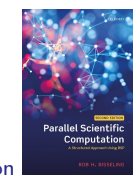
- Which distribution will yield this cost? All spreading assumptions must hold!



Square cyclic distribution? No!

- Previously, we have extolled the virtues of the square cyclic distribution for LU decomposition and all parallel linear algebra.
- Here, however, this distribution does not work well. Diagonal element a_{ii} is assigned to $P(i \bmod \sqrt{p}, i \bmod \sqrt{p})$, so that the matrix diagonal is assigned to the diagonal processors $P(s, s)$, $0 \leq s < \sqrt{p}$.
- Only $\sqrt{p}$ processors have part of the matrix diagonal and the vectors. The vector spreading assumption fails.
- The trouble is that diagonal processors must send $\sqrt{p} - 1$ copies of $\frac{n}{\sqrt{p}}$ vector components: $h_{\mathrm{s}} = n - \frac{n}{\sqrt{p}}$ in (0).
- The total cost for the square cyclic distribution is

$$T_{\text{MV, dense, } \sqrt{p}\times\sqrt{p} \text{ cyclic}} = \frac{2n^2}{p} + n + 2\left(1 - \frac{1}{\sqrt{p}}\right) ng + 4l.$$

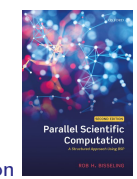


Cyclic row distribution? No!

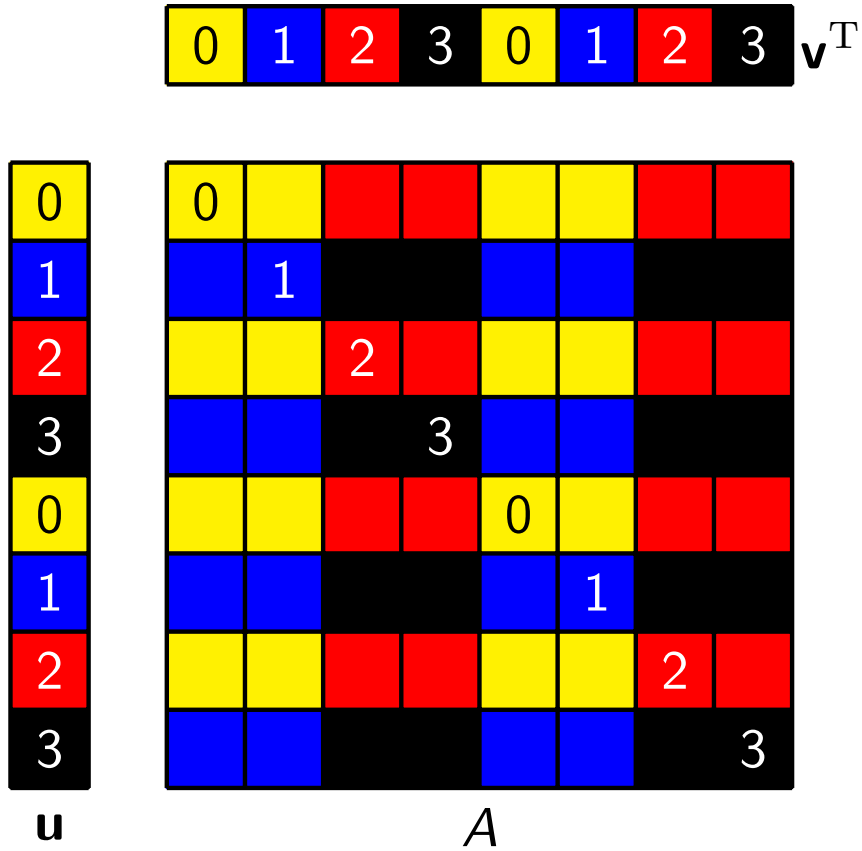
- The communication balance can be improved by choosing a distribution that spreads the matrix diagonal and the vectors evenly, $\phi_{\mathbf{u}}(i) = \phi_{\mathbf{v}}(i) = i \bmod p$, and translating the matrix distribution from 1D to 2D.
- We still have the freedom to choose M and N, where $MN = p$. For the choice $M = p$ and $N = 1$, this gives the cyclic row distribution $\phi_0(i) = i \bmod p$ and $\phi_1(j) = 0$.
- The total cost for the cyclic row distribution is

$$T_{\text{MV, dense, } p\times 1 \text{ cyclic}} = \frac{2n^2}{p} + \left(1 - \frac{1}{p}\right) ng + 2l.$$

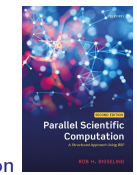
- This distribution skips supersteps (2) and (3), since each matrix row is completely contained in one processor.
- The trouble is that the fanout is very expensive: every processor has to send $\frac{n}{p}$ vector components to all others.



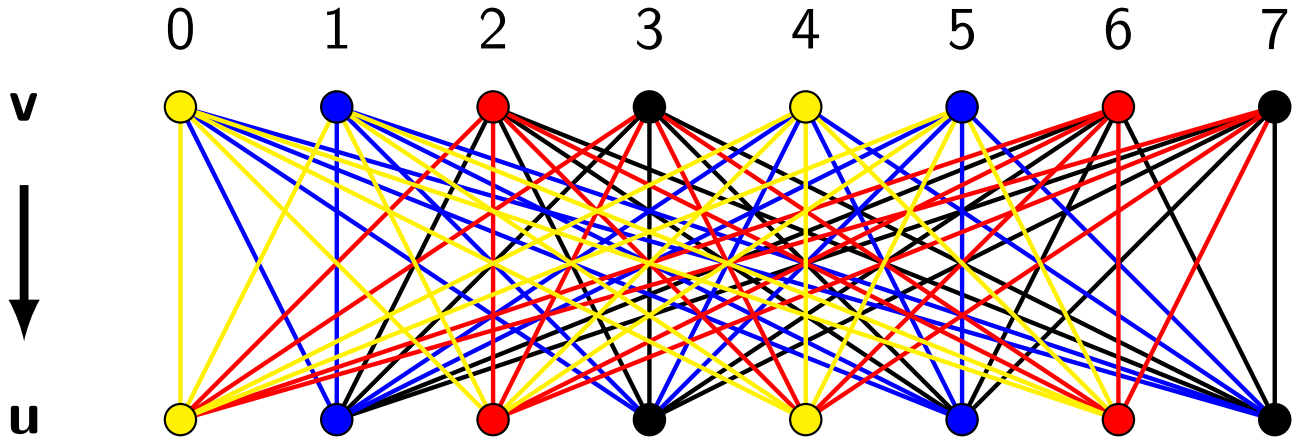
Square Cartesian distribution? Yes!



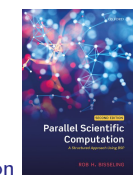
- Square Cartesian distribution based on a cyclic distribution of the matrix diagonal, $\phi_{\mathbf{u}}(i) = \phi_{\mathbf{v}}(i) = i \bmod p$, but now we choose $M = N = \sqrt{p}$ when translating from 1D to 2D.
- *Et voilà!* We achieve the optimal BSP cost.



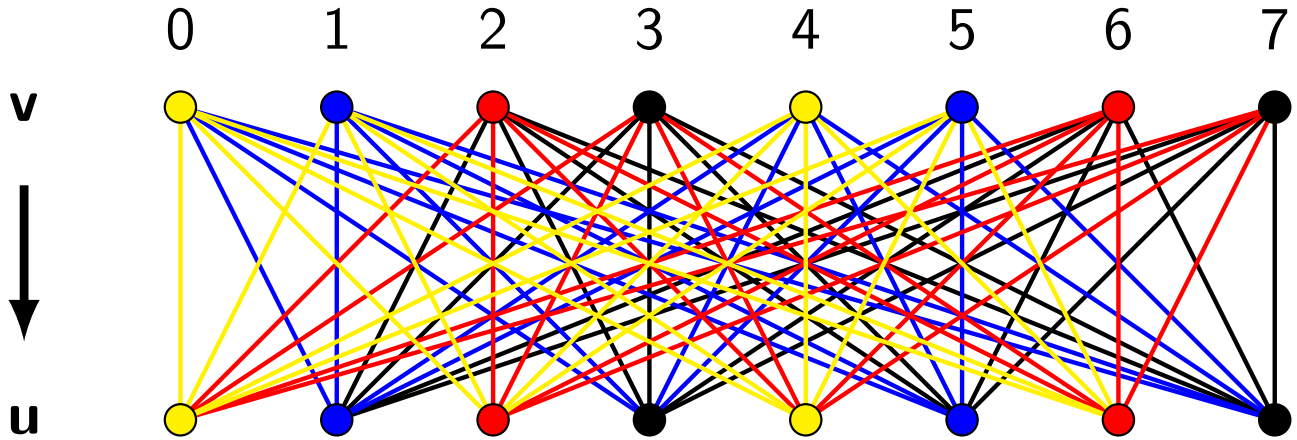
Two layers of a dense artificial neural network



- Each neuron in the top layer of the artificial neural network (ANN) is connected to all neurons of the bottom layer. Both layers have 8 neurons.
- The strength v_j of a signal fired by neuron j in the top layer is an input to the strength u_i of neuron i in the bottom layer.
- The connection between top neuron j and bottom neuron i carries a weight a_{ij} which has been determined during a training phase of the network.



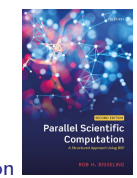
Similarity to matrix–vector multiplication



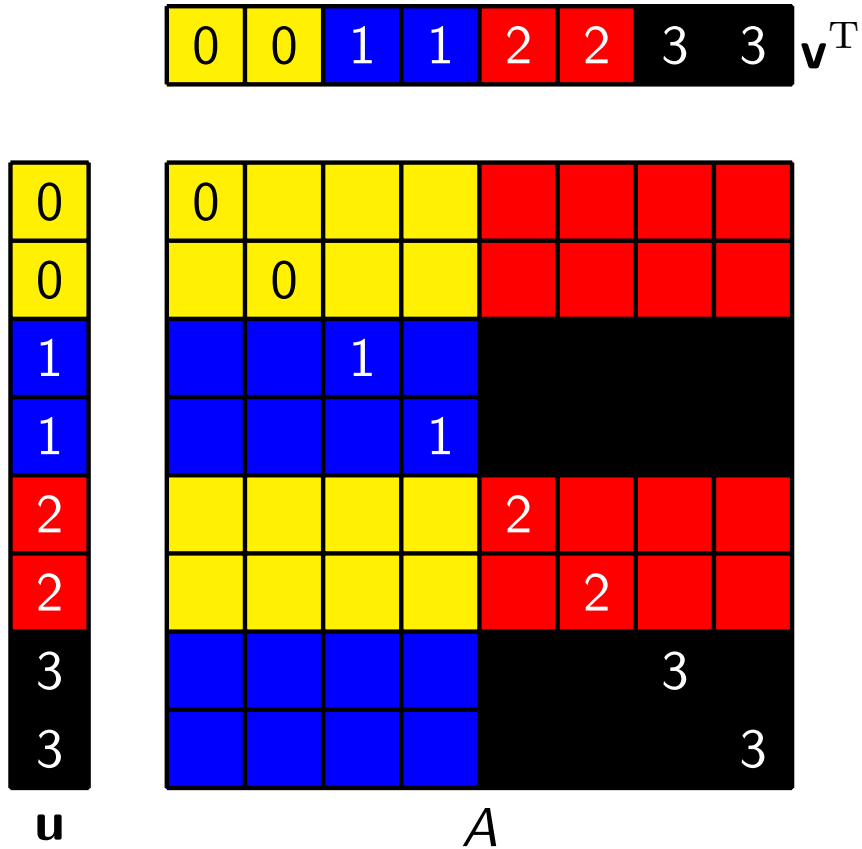
- The value u_i is typically given by a formula of the form

$$u_i = f\left(\sum_{j=0}^{n-1} a_{ij} v_j + b_i\right).$$

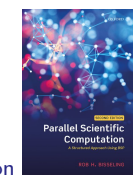
- Here, f is an activation function such as the Rectified Linear Unit (ReLU) function $f(x) = \max(0, x)$ and b_i is a component of a bias vector **b**.
- The neurons and their connections have been distributed over 4 processors according to the square Cartesian matrix distribution just discussed.



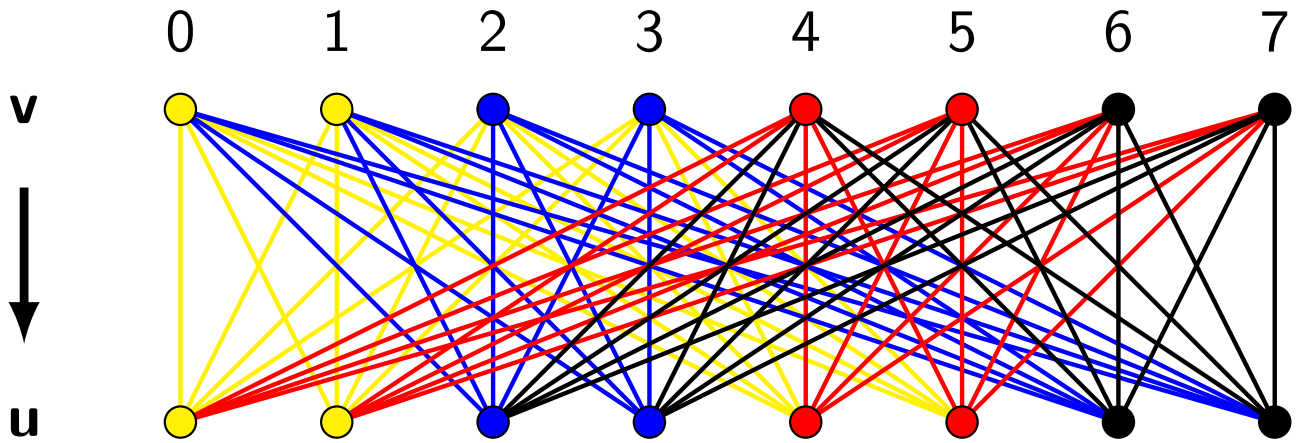
Square Cartesian distribution based on blocks



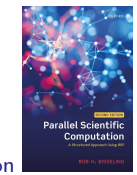
- Square Cartesian distribution based on a block distribution of the matrix diagonal, $\phi_{\mathbf{u}}(i) = \phi_{\mathbf{v}}(i) = i \text{ div } \lceil \frac{n}{p} \rceil$, with $M = N = \sqrt{p}$
- This also achieves the optimal BSP cost.



Distribution of artificial neural network based on blocks



- For dense ANNs, this distribution is just as good as the previous cyclic distribution.
- It has more locality in the picture.
- For sparse ANNs with locality, such as convolutional neural networks, a block-based distribution may be better.



Summary

- For Cartesian distributions, we use both 1D and 2D processor numberings to our advantage, with the identification

$$P(s,t) \equiv P(s+tM).$$

- We have seen the example of a tridiagonal matrix, where we obtained a 2D matrix distribution, slightly different from a 1D block row distribution.
- A square Cartesian matrix distribution based on a cyclic distribution of the matrix diagonal and the vectors is an optimal data distribution for dense matrices and for sparse matrices that are relatively dense.
- This distribution (or a block-based alternative) can also be applied to the neurons and the weights of a dense artificial neural network.