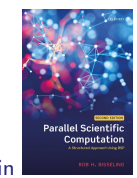
Fine-Grain and Medium-Grain Matrix Distribution

Section 4.6 of Parallel Scientific Computation, 2nd edition

Rob H. Bisseling

Utrecht University



Fine-grain model

- Create a hypergraph from an $m \times n$ sparse matrix A where each nonzero $a_{ij} \neq 0$ becomes a vertex.
- Each row i gives rise to a row net

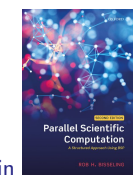
$$\{j \ : \ 0 \leq j < n \ \wedge \ a_{ij} \neq 0\}.$$

- Each column j gives rise to a column net

$$\{i \ : \ 0 \leq i < m \ \wedge \ a_{ij} \neq 0\}.$$

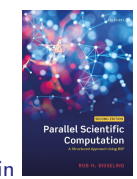
- The hypergraph has $nz(A)$ vertices and $m + n$ nets.

Ü. V. Çatalyürek and C. Aykanat, In: Proceedings Irregular 2001, IEEE, p.118.



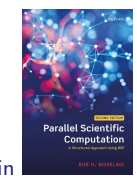
Fine-grain partitioning

- $\lambda_i = \#$ processors with a nonzero in row i
  $= \#$ processors having a vertex in row net i.
- $\lambda_i - 1 =$ communication volume caused by row net i.
- $\mu_j - 1 =$ communication volume caused by column net j.
- Partitioning the fine-grain hypergraph minimizes the exact communication volume of the parallel SpMV.
- Partitioning can be expensive since the number of vertices is large.



Desirables

- We would like to keep the nonzeros of each row together.
- We would like to keep the nonzeros of each column together.
- We would like to have our cake and eat it too.



Medium-grain partitioning

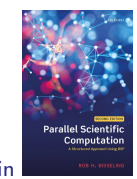
- Solution: split the matrix A by a simple method into

$$A = A^{\mathrm{r}} \cup A^{\mathrm{c}} = A^{\mathrm{r}} + A^{\mathrm{c}},$$

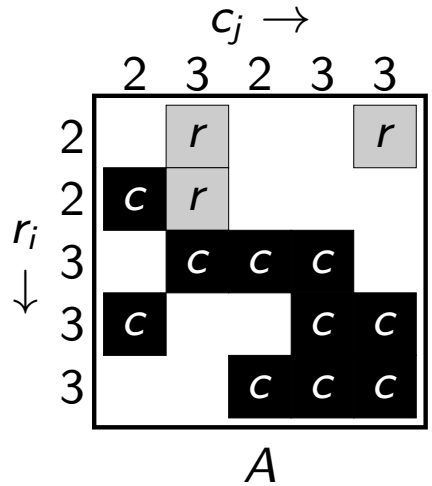
  where $A^{\mathrm{r}} \cap A^{\mathrm{c}} = \emptyset$.
- The nonzeros in a row of A^{r} stay together and those in a column of A^{c} also stay together.
- For a square matrix A, form the $2n \times 2n$ matrix

$$B = \begin{bmatrix} I_n & (A^{\mathrm{r}})^{\mathrm{T}} \\ A^{\mathrm{c}} & I_n \end{bmatrix}.$$

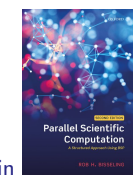
D. M. Pelt and R. H. Bisseling, In: Proceedings IPDPS 2014, IEEE, pp. 529–539.



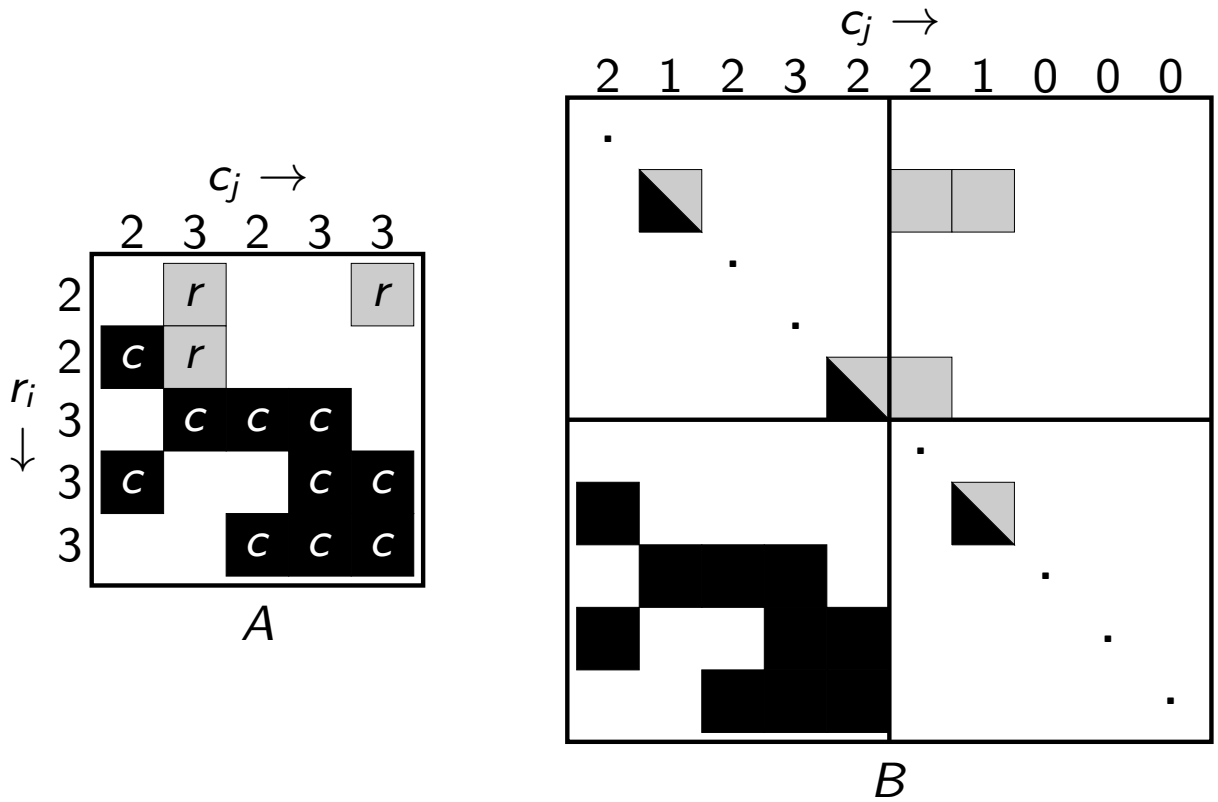
Simple split of matrix A



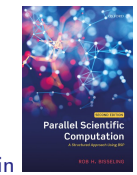
- The split is based on
  - $r_i = \#$ nonzeros of row i
  - $c_j = \#$ nonzeros of column j.
- Nonzero a_{ij} is assigned to A^{r} if $r_i < c_j$, because rows with fewer nonzeros are more likely to stay together.
- Exception: if $r_i = 1$, the row cannot be cut, so a_{ij} is assigned to A^{c} to help keep its column together.
- If $c_j < r_i$, the nonzero is assigned to A^{c}.
- All ties a_{ij} with $r_i = c_j$ are broken in the same way.



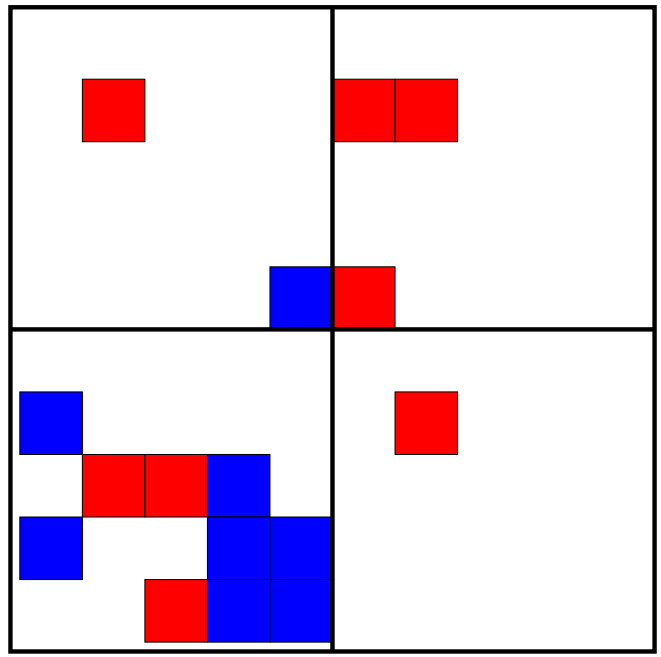
Form new matrix B



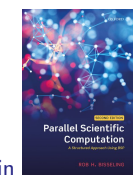
- The diagonal entries in the top left block B connect the nonzeros originating in the same column of A.
- Unnecessary diagonal entries caused by empty rows in A^{r} are removed from B.



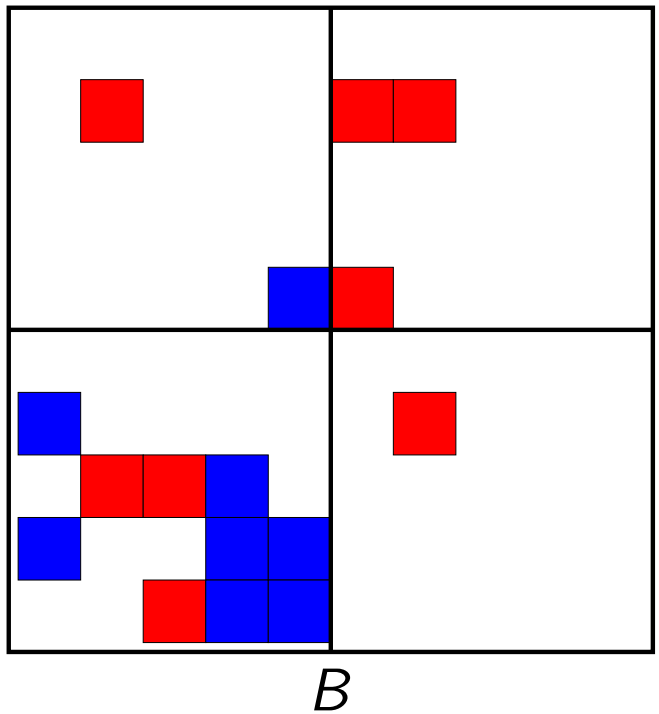
Bipartition matrix B by columns

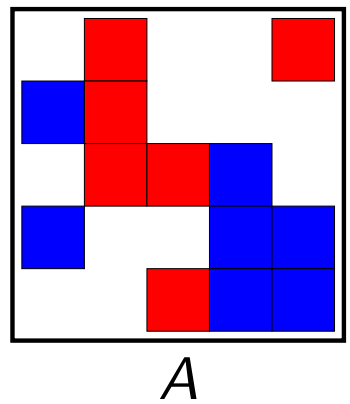


- Perform a 1D column-wise bipartitioning of B.
- The allowed imbalance $\epsilon = 0.1$, where the diagonal nonzeros do not count.
- Resulting communication volume $V(B_0, B_1) = 4$.



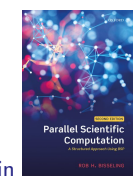
Fold B back into A



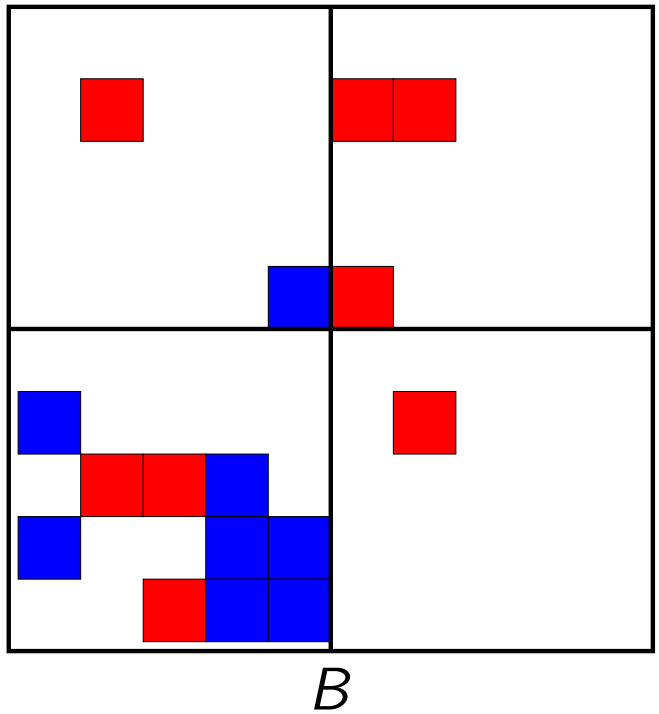


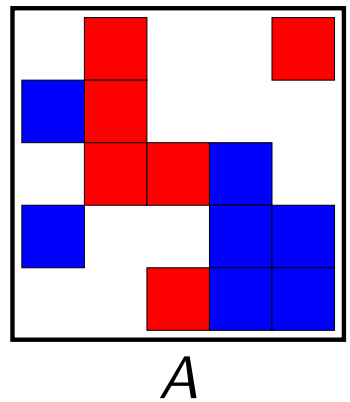
- For nonempty rows j of B: the number of processors in column j of A equals the number of processors in row j of B,

$$\mu_j(A_0, A_1) = \lambda_j(B_0, B_1), \quad \text{for } 0 \leq j < n.$$

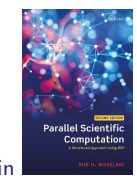


Fold B back into A





- Look at the last column of A, column $j = 4$: its nonzeros come from column 4 of B and row 4 of B.
- But by construction, column 4 of B has a single owner, the owner of b_{44}, which already occurs in row 4 of B.
- So row 4 of B determines the amount of communication in column 4 of A.



Communication volume is preserved

- We have proven that

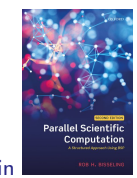
$$\mu_j(A_0, A_1) = \lambda_j(B_0, B_1), \quad \text{for } 0 \leq j < n.$$

- Similarly, we can prove that

$$\lambda_i(A_0, A_1) = \lambda_{n+i}(B_0, B_1), \quad \text{for } 0 \leq i < n.$$

- Therefore, using the notation $\lambda'_i = \max(\lambda_i, 0)$,

$$\begin{aligned} V(A_0, A_1) &= \sum_{i=0}^{n-1} \lambda'_i(A_0, A_1) + \sum_{j=0}^{n-1} \mu'_j(A_0, A_1) \\ &= \sum_{i=0}^{n-1} \lambda'_{n+i}(B_0, B_1) + \sum_{j=0}^{n-1} \lambda'_j(B_0, B_1) \\ &= \sum_{i=0}^{2n-1} \lambda'_i(B_0, B_1) = V(B_0, B_1). \end{aligned}$$



Regular 1D partitioning is a special case

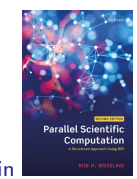
- If we split a square $n \times n$ matrix A into $A = A^{\mathrm{r}} \cup A^{\mathrm{c}}$ by choosing $A^{\mathrm{r}} = 0$ and $A^{\mathrm{c}} = A$, we obtain the $2n \times 2n$ matrix

$$B = \begin{bmatrix} 0 & 0 \\ A & 0 \end{bmatrix}.$$

- Here, the matrix diagonal became completely empty after removal of unnecessary entries.
- A 1D column-wise partitioning of B then reduces to a 1D column-wise partitioning of A.
- For the choice $A^{\mathrm{r}} = A$ and $A^{\mathrm{c}} = 0$, the matrix B becomes

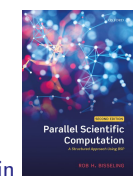
$$B = \begin{bmatrix} 0 & A^{\mathrm{T}} \\ 0 & 0 \end{bmatrix}.$$

- A 1D column-wise partitioning of B then reduces to a 1D row-wise partitioning of A.

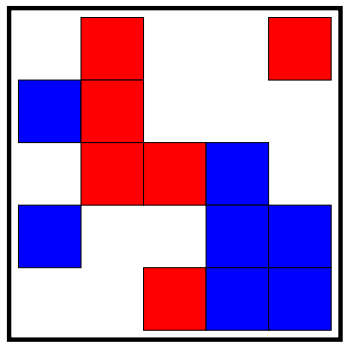


Chicken-or-egg problem: which one was first?

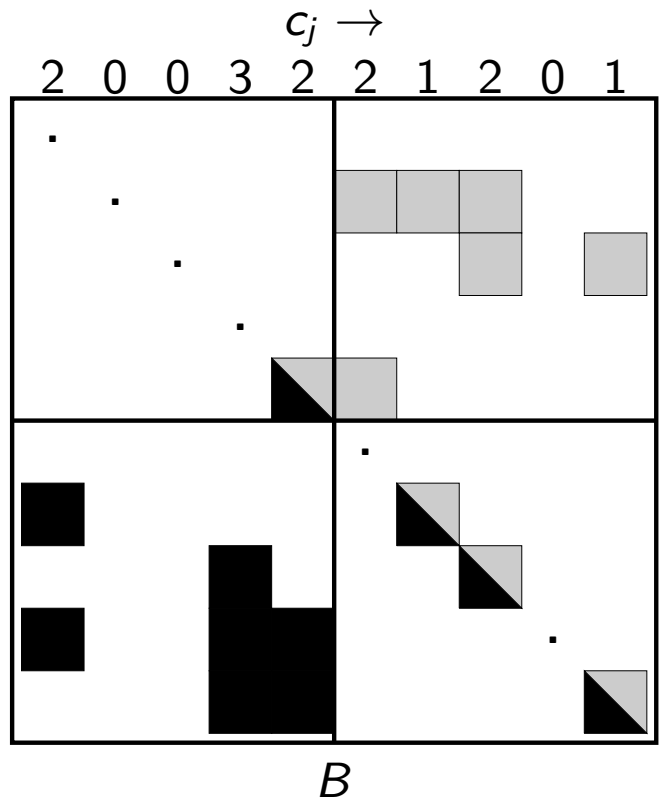
- To partition the matrix A, we first need to form a matrix B.
- To form a matrix B, we need a partitioning of A.
- That's why we start with a simple partitioning of A.



Iterative refinement by repeated partitioning

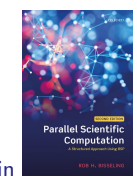


$A = A^r + A^c$



B

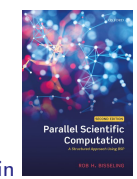
- Iterative refinement uses the output of a partitioning as input to a next partitioning: $A^r = A_0$ and $A^c = A_1$.
- The next partitioning consists of 1 level of Kernighan–Lin refinement, which is fast and can only improve the result.



Test set of sparse square matrices

Name	n	nz	c	Origin
mip1	66 463	10 352 819	155.8	mixed integer programming
in-2004	1 382 908	16 917 053	12.2	web links India 2004
asia_osm	11 950 757	25 423 206	2.1	road network Asia
cage14	1 505 785	27 130 349	18.0	DNA electrophoresis
rgg_n_2_21_s0	2 097 152	28 975 990	13.8	random geometric graph

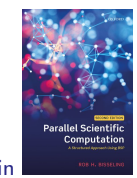
- Matrices from the SuiteSparse Matrix (formerly University of Florida) Collection by Tim Davis.



Communication volume and partitioning time

Name	p	Volume			Time (in s)		
		LB	FG	MG	LB	FG	MG
mip1	2	9 099	3 929	2 109	94	291	98
	64	120 636	90 133	56 864	350	1 059	230
in-2004	2	1 158	637	558	81	376	89
	64	18 247	16 345	14 425	401	1 774	397
asia_osm	2	91	120	130	61	48	48
	64	2 291	2 667	2 538	271	206	258
cage14	2	195 912	172 091	154 962	153	232	109
	64	1 436 410	1 161 269	980 957	664	1 035	516
rgg_n_2_21_s0	2	3 364	3 322	2 976	47	111	64
	64	46 192	44 049	41 249	234	613	345
Norm. geomean		1.00	0.83	0.70	1.00	2.10	0.95

- LB = Localbest (original Mondriaan)
- FG = Fine-grain
- MG = Medium-grain with iterative refinement



Geometric mean

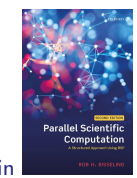
- Definition for a set of values $x_0, \ldots, x_{k-1} > 0$:

$$GM(x_0, \ldots, x_{k-1}) = (x_0 \cdot x_1 \cdot \cdots \cdot x_{k-1})^{\frac{1}{k}}.$$

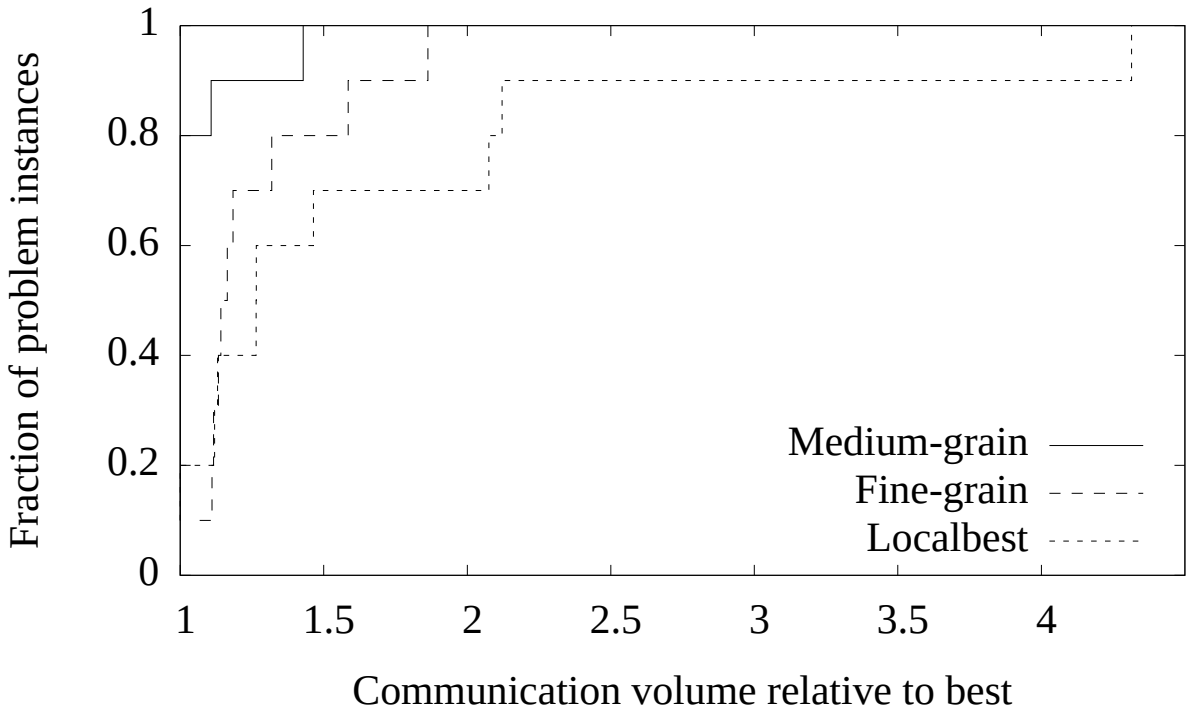
- GM can handle widely differing scales.
- It gives each matrix/p pair equal influence.
- Useful property:

$$GM(\frac{x_0}{y_0}, \ldots, \frac{x_{k-1}}{y_{k-1}}) = \frac{GM(x_0, \ldots, x_{k-1})}{GM(y_0, \ldots, y_{k-1})}.$$

- It does not matter whether we normalize the values first or only after computing the mean.
- This is useful when comparing results from different methods, normalizing against one method.

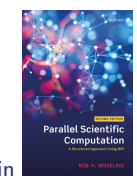


Performance profile of the communication volume



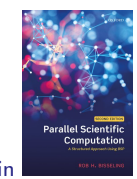
Higher is better!

- How to read this: the Localbest method solves 70% of the 10 problem instances within 1.5 times the lowest volume achieved by any of the three methods.
- The medium-grain method solves all problem instances within 1.5 times the lowest volume.



Advantages of performance profiles

- Problem instances for which one method fails but others succeed can still be included in a performance profile (in contrast to geometric-mean comparisons).
- A performance profile can be used to summarize results for a large number of problem instances.
- Example: all the 2833 matrices from the SuiteSparse Matrix collection partitioned for $p = 2, 4, 8, \ldots, 1024$.
- A performance profile captures much more information than a single number such as the geometric mean.



Summary

- The fine-grain method can in principle find the best partitioning of a given sparse matrix.
- The medium-grain method, however, usually achieves this in practice.
- The medium-grain method tries to keep both rows and columns together, based on a simple nonzero-count criterion.
- This is in contrast to the original Mondriaan method (Localbest), which imposes both objectives separately and then takes the best result.
- The geometric mean and performance profiles are useful in comparing different methods and summarizing their results.