

Parallel Computation of Bruhat Decomposition in the BSP Model

Ioanna Evtushevskaya-Konovalova
Alexander Tiskin

St.Petersburg University

September 29, 2026

Outline

- 1 Introduction
- 2 BSP Parallel Computing Model
- 3 Block-Recursive linear algebra
- 4 Bruhat Decomposition Algorithm
- 5 Complexity Analysis
- 6 Antidiagonal Recursion

LU-Decomposition

A – non-singular square matrix, all leading principal minors are non-singular.

LU-decomposition – $A = LU$

U – non-singular upper triangular matrix

L – non-singular lower triangular matrix

LU Decomposition: Applications to Computational Problems

$$\begin{cases} A = LU \\ Ax = b \end{cases} \Rightarrow LUx = b \Leftrightarrow \begin{cases} Ly = b \\ Ux = y \end{cases}$$

Bruhat Decomposition

Bruhat decomposition is a natural generalization of LU decomposition to the case where the leading minors of the matrix may be singular.

Bruhat decomposition

A – non-singular square matrix

Bruhat decomposition – $A = U_1 P U_2$

U_1, U_2 – non-singular upper triangular matrices

P – permutation matrix.

Applications of Bruhat Decomposition to Computational Problems

Suppose $A = U_1 P U_2$ is a Bruhat decomposition. Then

$$Ax = b \Leftrightarrow (U_1 P U_2)x = b \Leftrightarrow \begin{cases} U_1 y = b \\ P U_2 x = y \end{cases} \Leftrightarrow \begin{cases} U_1 y = b \\ U_2 x = P^T y \end{cases}$$

Bruhat Decomposition in Abstract Algebra

Bruhat decomposition is actively used in abstract algebra.

Bulk Synchronous Parallelism

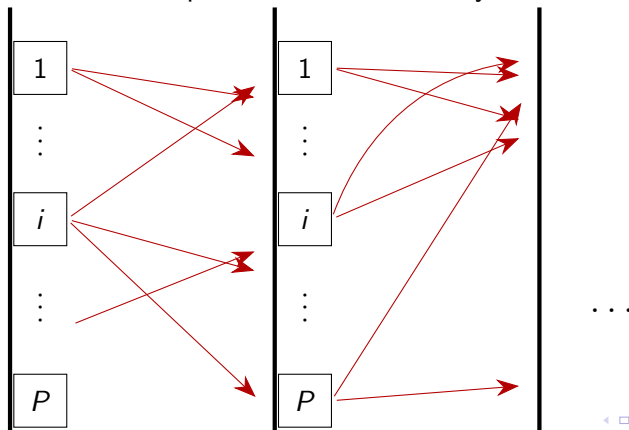
The BSP model was proposed by Leslie Valiant in 1990.

- p – model parameter, number of processors
- each processor has local memory, and there is shared global memory accessible to all processors
- processors can perform computations in parallel and exchange information with each other

- the network routes information exchange between pairs of processors
- g is a model parameter representing the time required to send or receive one unit of information.

Synchronization

- an algorithm is a sequence of supersteps. During a superstep, computations are asynchronous and processors cannot receive data from each other. At the end of each superstep, barrier synchronization occurs and data is exchanged between processors.
- l - model parameter, the cost of synchronization.



Computational Cost

p, g, l – main system's parameters

n – the size of the input data.

Algorithm – sequence of supersteps, its number – $S(n, p)$.

So:

k – superstep number

r – processor number

$W_{k,r}(n, p)$ – computation cost on processor r during superstep k .

$H_{k,r}(n, p)$ – communication cost on processor r during superstep k .

The computation cost per superstep k :

$$W_k(n, p) = \max_r (W_{k,r}(n, p))$$

The communication cost per superstep k :

$$H_k(n, p) = \max_r (H_{k,r}(n, p)).$$

Total cost:

$$\sum_{i=0}^{S(n,p)} (W_i(n, p) + g \cdot H_i(n, p) + l).$$

Block-Recursive linear algebra

In this work, a block-recursive approach is used for parallel computation.

- the classical Strassen algorithm for matrix multiplication
- The fastest (currently known) matrix multiplication algorithm (Vassilevska Williams)
- Fast Fourier transform
- Actively used for matrix computation algorithms.
- Useful libraries are being created in some programming languages.

- 3D algorithm matrix multiplication in the BSP model can be performed with the following costs:

$$W = O\left(\frac{n^3}{p}\right) \quad (1)$$

$$H = O\left(\frac{n^2}{p^{2/3}}\right) \quad (2)$$

$$S = O(1) \quad (3)$$

Matching lower bound on the computation cost:

$$W = O\left(\frac{n^3}{p}\right)$$

(McColl, Tiskin, 1999)

Matching lower bound on the communication cost:

Consider parallel multiplication of $n \times n$ matrices, p processors.

Then:

$$H(n, p) \geq \frac{n^2}{p^{2/3}} \quad (4)$$

(Based on Loomis–Whitney inequality)

(Irony, Toledo, Tiskin, 2004.)

LU decomposition with pivoting pairwise can also be performed in parallel in the BSP model with the following costs:

$$W = O\left(\frac{n^3}{p}\right)$$

$$H = O\left(\frac{n^2}{p^{2-\log_2 3}}\right) \approx O\left(\frac{n^2}{p^{0.42}}\right)$$

$$S = O(3^k) = O(p^{\log_2 3}) \approx O(p^{1.58})$$

(2007, Tiskin)

$$W = O\left(\frac{n^3}{p}\right)$$

$$H = O\left(\frac{n^2}{p^{2/3}}\right)$$

$$S = O(3^k) = O(p^{2/3})$$

Bruhat decomposition

A – non-singular square matrix

Bruhat decomposition – $A = U_1 P U_2$

U_1, U_2 – non-singular upper triangular matrices

P – permutation matrix.

A more convenient formulation:

U - upper triangular

L - lower triangular

P - permutation matrix

and the following holds:

$$LAU = P \quad (5)$$

LEU-decomposition

Matrix A – square matrix, maybe singular

I – set of zero rows number A

J – set of zero columns number A

LEU-decomposition – $A = LEU$

U – non-singular upper-triangular matrix, L – non-singular lower-triangular matrix

E – subpermutation matrix: rows E in I and columns E in J are zero, $rk(E) = rk(A)$.

“Fast computation of the rank profile matrix and the generalized Bruhat decomposition”, Dumas, Pernet, Sultan

“Fast generalized Bruhat decomposition”, Malashonok.

Square-recursive algorithm

The algorithm takes a matrix B as input and processes it in blocks, in several stages.

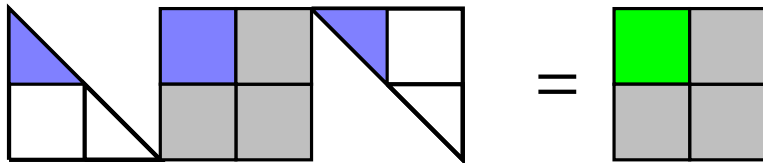
$$B = \begin{pmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{pmatrix} \quad (6)$$

Recursive Call I

1. LEU -decomposition for b_{00}

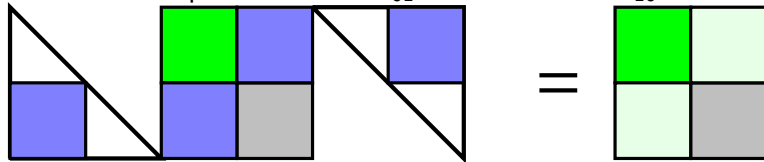
$$l_{00} b_{00} u_{00} = p_{00} \quad (7)$$

$$L_1 = \begin{pmatrix} l_{00} & 0 \\ 0 & e \end{pmatrix} U_1 = \begin{pmatrix} u_{00} & 0 \\ 0 & e \end{pmatrix} \quad (8)$$



Preprocessing of Anti-Diagonal Blocks

2. Elimination part of rows of b_{01} and columns of b_{10} :



$$L' = \begin{pmatrix} e & 0 \\ l' & e \end{pmatrix} U' = \begin{pmatrix} e & u' \\ 0 & e \end{pmatrix} \quad (9)$$

where

$$l' = -(b_{(10)})p_{00}^T \quad (10)$$

$$u' = -(b_{(01)})p_{00}^T \quad (11)$$

$$B = \begin{pmatrix} p_{00} & b_{(01)}^* \\ b_{(10)}^* & b_{(11)}' \end{pmatrix}$$

Recursive Calls II and III

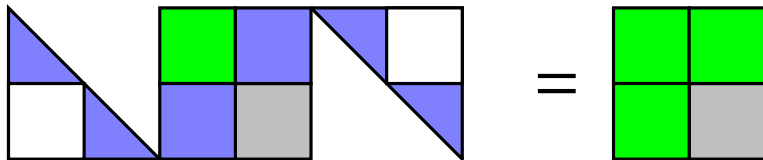
3. *LEU*-decomposition for b_{01}^* and b_{10}^*

$$l_{01}^* b_{01}^* u_{01}^* = p_{01}^* \quad (12)$$

$$l_{10}^* b_{10}^* u_{10}^* = p_{10}^* \quad (13)$$

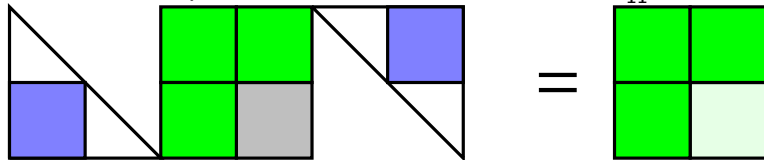
$$L_2 = \begin{pmatrix} l_{01}^* & 0 \\ 0 & l_{10}^* \end{pmatrix} U_2 = \begin{pmatrix} u_{01}^* & 0 \\ 0 & u_{10}^* \end{pmatrix}$$

(14)



Preprocessing of Block b_{11}^*

4. Elimination part of rows and columns of block b_{11}^*



$$L'' = \begin{pmatrix} e & 0 \\ l'' & e \end{pmatrix} U'' = \begin{pmatrix} e & u'' \\ 0 & e \end{pmatrix} \quad (15)$$

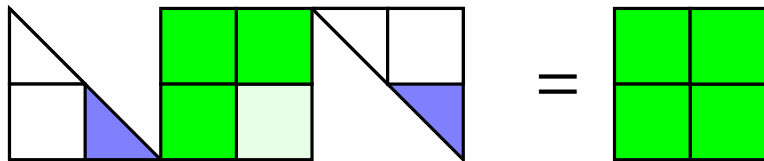
$$l'' = -b'_{11} p_{01}^T \quad (16)$$

$$u'' = -b'_{11} p_{10}^T \quad (17)$$

5. *LEU*-decomposition for the updated b_{11}^*

$$l_{11}^* b_{11}^* u_{11}^* = p b_{11}^* \quad (18)$$

$$L_3 = \begin{pmatrix} e & 0 \\ 0 & l_{11}^* \end{pmatrix} U_2 = \begin{pmatrix} e & 0 \\ 0 & u_{11}^* \end{pmatrix} \quad (19)$$



Thus, the matrix B is reduced to a sub-permutation matrix by left multiplication by lower triangular matrices and right multiplication by upper triangular matrices, which is equivalent to finding the (generalized) Bruhat decomposition:

$$L_1 L' L_2 L'' L_3 B U_3 U'' U_2 U' U_1 = P \quad (20)$$

Parallel matrix multiplication:

$$W(n, p) = O(n^3/p)$$

$$H(n, p) = O(n^2/p^{2/3})$$

$$S(n, p) = O(1)$$

$$W(n, p) = O\left(\frac{n^3}{p^{\alpha(3-\log 3)}}\right) + O\left(\frac{n^3}{p}\right) = O\left(\frac{n^3}{p}\right)$$

$$H(n, p) = \begin{cases} O\left(\frac{n^2}{p^{2/3}}\right) & \alpha \geq \frac{2}{3(2-\log 3)} = \alpha_{\max} \approx 1.59 \\ O\left(\frac{n^2}{p^{\alpha(2-\log 3)}}\right) & \text{otherwise} \end{cases}$$

$$S(n, p) = O(3^{k_0}) \approx O(p^{\alpha 1.58})$$

$$0.70 \approx 1/(3 - \log 3) = \alpha_{\min} \leq \alpha \leq \alpha_{\max} \approx 1.59$$

Banded Matrices: Definition

Definition

Matrix A is called **lower banded of bandwidth k** if

$$A_{ij} \neq 0 \Rightarrow 0 \leq i - j \leq k.$$

Properties

- Nonzero entries appear only in a strip of width k below the main diagonal.
- Upper banded matrices are defined analogously (strip above the diagonal).

Banded Matrices: Multiplication Lemma

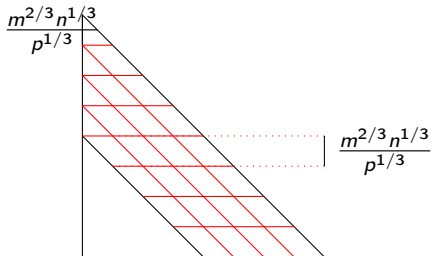
Lemma

Let matrices A, B be lower banded of bandwidth k . Then the product AB is lower banded of bandwidth $2k$.

Lemma

Let A, B be $n \times n$ matrices, lower banded of bandwidth m . Their product can be computed in parallel on p processors with costs:

$$W = O\left(\frac{m^2 n}{p}\right), \quad H = O\left(\frac{m^{4/3} n^{2/3}}{p^{2/3}}\right), \quad S = O(1).$$



Reducing an Antidiagonal Strip

We will say that a matrix (or a submatrix) is **reduced** if it contains at most one nonzero entry in any row and any column, and that entry is equal to 1.

Lemma

An antidiagonal strip of width n_0 of an $n \times n$ matrix can be reduced by multiplying on the left and right by banded matrices of bandwidth n_0 . The cost is

$$W = O(n_0^3), \quad H = O(n_0^2).$$

Elimination of Rows and Columns

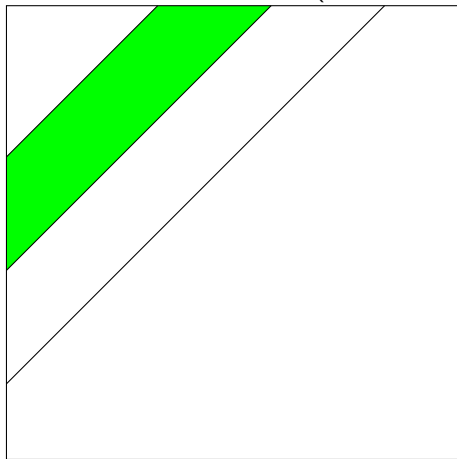
Lemma

Suppose the first m antidiagonals have been reduced, containing nonzero elements in columns in J and rows in I . By multiplying on the left by a lower-banded matrix L and on the right by an upper-banded matrix U (both bandwidth $2m$), we can zero out the intersection of these rows/columns with antidiagonals m through $2m$.

Strip-recursive algorithm

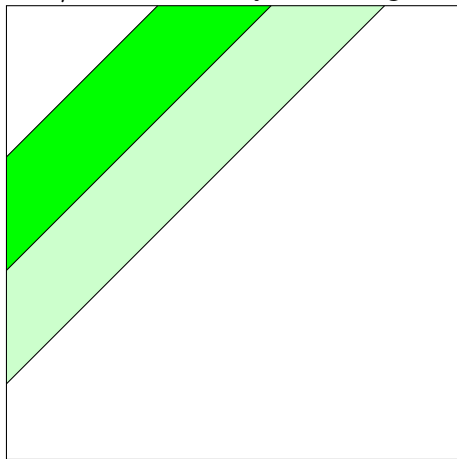
Input: a strip of width $m = d \cdot n_0$ (several consecutive antidiagonals).

1. **Recursive call** on the upper-left half-strip (width $m/2$). Multiply by banded matrices L_m, U_m (bandwidth $2m$).



Strip-recursive algorithm

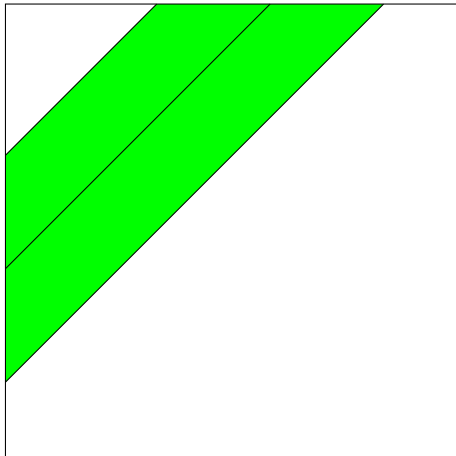
2. **Update** the antidiagonal half-strip (also width $m/2$). Multiply by L_m , U_m ; compute preprocessing matrices L'_m , U'_m (bandwidth $2m$) to eliminate rows/columns already containing nonzeros in the upper-left half.



Strip-recursive algorithm

Input: a strip of width $m = d \cdot n_0$ (several consecutive antidiagonals).

3. **Second recursive call** on the lower-right half-strip, obtaining $\tilde{L}_m$, $\tilde{U}_m$.



4. Combine:

$$L_{\text{Fin}} = \tilde{L}_m \cdot L'_m \cdot L_m, \quad U_{\text{Fin}} = \tilde{U}_m \cdot U'_m \cdot U_m.$$

$$W = O\left(\frac{n^3}{p}\right) + O\left(\frac{n^3}{p^{2\alpha}}\right) = O\left(\frac{n^3}{p}\right)$$

$$H = \begin{cases} O\left(\frac{n^2}{p^{\frac{2}{3}}}\right) & \alpha \geq \alpha_{\max} = \frac{2}{3} \\ O\left(\frac{n^2}{p^{\alpha}}\right) & \text{otherwise} \end{cases}$$

$$S = O(p^{\alpha})$$

$$1/2 = \alpha_{\min} \leq \alpha \leq \alpha_{\max} = 2/3$$

Conclusion

So, we developed and compared two algorithms in *BSP* model based on different recursion schemes and analyzed them.

The antidiagonal recursion scheme is **more efficient** than the square-block recursion scheme.

It achieves **optimal** computational and communication costs.

So, the strip-recursive algorithm is more promising for implementation, especially for machines with high synchronization costs. However, it has a more complex partition structure.

- ❶ Jean-Guillaume Dumas, Clément Pernet, and Ziad Sultan. Fast computation of the rank profile matrix and the generalized Bruhat decomposition. *Journal of Symbolic Computation*, 83:187–210, 2017.
- ❷ D.Yu. Grigor'ev. Additive complexity in directed computations. *Theoretical Computer Science*, 19(1):39–67, 1982.
- ❸ Gennadi Malaschonok. Fast Generalized Bruhat Decomposition. pp. 194–202, 2010.
- ❹ W. F. McColl and A. Tiskin. Memory-Efficient Matrix Multiplication in the BSP Model. *Algorithmica*, 24(3-4):287–297, 1999.
- ❺ Alexander Tiskin. Communication-efficient parallel generic pairwise elimination. *Future Generation Computer Systems*, 23:179–188, 2007.
- ❻ L. G. Valiant. A Bridging Model for Parallel Computation. *Communications of the ACM*, 33(8):103–111, 1990.