Project Assignment I: All-Pairs Shortest Path (APSP)

Repeated Squaring + Fox (MPI)

Sérgio Cardoso up202107918
Kathleen Soares up201903010

October 27, 2025

Abstract

This report describes the parallel implementation of the *All-Pairs Shortest Path* (APSP) problem through the min-plus product and the *Repeated Squaring* method, using Fox's algorithm over MPI. The program was developed in C, with a distributed approach based on a grid of processes $Q \times Q$, where $P = Q^2$. The main design decisions, communication and data distribution details, as well as preliminary performance measurements are presented.

1 Introduction

The objective of this project is to determine the minimum distances between all pairs of vertices in a weighted directed graph, represented by an adjacency matrix A. Each entry A_{ij} contains the weight of the edge (i, j), or an infinite value when there is no direct connection. The algorithm uses the min-plus product and the *Repeated Squaring* method, with each matrix multiplication implemented in parallel via Fox's algorithm in MPI.

2 Algorithmic Base

2.1 Product Min-Plus

The product min-plus between two matrices A and B is defined by:

$$C_{ij} = \min_k \bigl(A_{ik} + B_{kj}\bigr). \tag{1}$$

This operation replaces the traditional sum with the minimum, and the multiplication with the sum. Thus, the distance matrices are multiplied in such a way as to propagate minimum paths.

On this operation, the traditional sum is replaced by the minimum and the multiplication by the sum. Thus, the traditional operations of linear algebra — multiplication and sum — are replaced by addition and minimization, allowing the propagation of the smallest distances in a weighted graph.

In the context of this project, this same logic is implemented in the function `Local_matrix_minplus()` of the C code, where each MPI process locally performs the calculation of a block of C from blocks of A and B. This operation constitutes the core of Fox's algorithm, used to perform the distributed multiplication of matrices in the method of *Repeated Squaring*, ensuring the obtaining of the smallest distances between all pairs of vertices.

2.2 Repeated Squaring

The technique of *Repeated Squaring* consists of repeating the min-plus product of a matrix by itself ($A^2, A^4, A^8, \ldots$) until the distances do not change or the number of iterations is sufficient $\lceil \log_2 N \rceil$. In the code, this repetition is controlled by a loop in which, at each iteration, the function `Fox()` is called.

2.3 Fox's Algorithm

The Fox's algorithm is utilized to multiply blocks of matrices in parallel. The matrix is divided into sub-blocks $(N/Q) \times (N/Q)$, distributed among the processes arranged in a Cartesian grid. Each process executes, per phase, the diffusion of blocks of A along the row and a circular rotation of the blocks of B along the column, computing the local part of C with the min-plus operation.

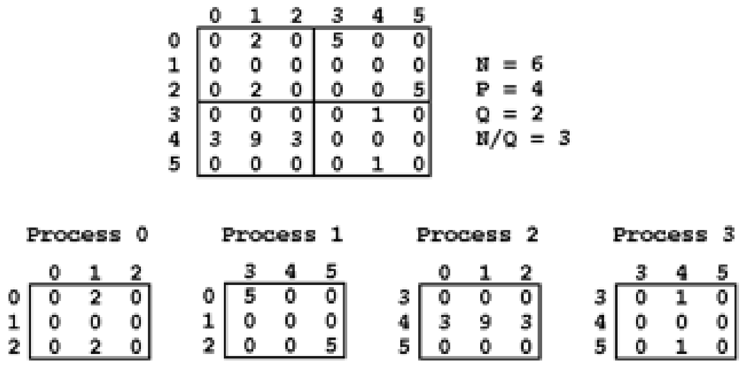

Figure 1: Speedup of the Fox algorithm for different matrix sizes.

3 Implementation

3.1 Data Structures

They were defined two main data types:

- **GRID_INFO_TYPE:** Structure that stores the topology of MPI processes (grid, communicators by row and column, dimensions, and identifiers of each process);
- **LOCAL_MATRIX_TYPE:** Structure that represents a local block of the matrix, with size $n_b = N/Q$ and a pointer to the dynamically allocated data.

Each process maintains local copies of three matrices: `A_local`, `B_local` and `C_local`. The infinite value is represented by `INT_MAX` = 10^9, and the diagonal is forced to zero.

3.2 Data Distribution

The input matrix is read only by process 0, which splits it into blocks of size $(n_b \times n_b)$ and distributes them to the remaining processes using `MPI_Send` and `MPI_Recv`. This process implements a 2D block distribution, ensuring that each process has the corresponding sub-block of the original matrix.

3.3 Communication and Synchronization

The function `Setup_grid()` creates the process grid with `MPI_Cart_create`, as well as specific communicators for rows and columns. In the `Fox()` function, each iteration executes:

1. Block diffusion of A along the row with `MPI_Bcast`;
2. Local min-plus multiplication between blocks of A and B;
3. Update of the local block C with the minimum between the current value and the new;
4. Rotation of the blocks of B using `MPI_Sendrecv_replace()`.

3.4 Repeated Squaring and Convergence

Multiplication is repeated $\lceil \log_2 N \rceil$ times. In each iteration, `Fox()` is called with the same input blocks ($A = B$), and the result is swapped between the variables `src` and `dst`. The program terminates when the iterations are completed, producing the final result in `src`.

3.5 Input and Output

Process 0 reads the array from `stdin`, substituting 0 and -1 with `INT_MAX`. After the iterations, the result blocks are gathered with `MPI_Recv` and `MPI_Send` back to process 0, which prints the final matrix. Entries without a path are returned as -1.

4 Results and Evaluation

5 Partial Conclusions

The program implements the foundations of the min-plus product and Fox's algorithm in MPI, combined to solve the APSP via *Repeated Squaring*.

References

[1] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.

[2] Fox, G. C., Otto, S. W., & Hey, A. J. (1987). *Matrix algorithms on a hypercube I: Matrix multiplication.* Parallel Computing, 4(1), 17–31.

[3] Pacheco, P. S. (1998). *A User's Guide to MPI.* San Francisco: Department of Mathematics, University of San Francisco.

Note: The complete source code (`cp_projecti.c`) is attached in the ZIP file submitted along with this report.