

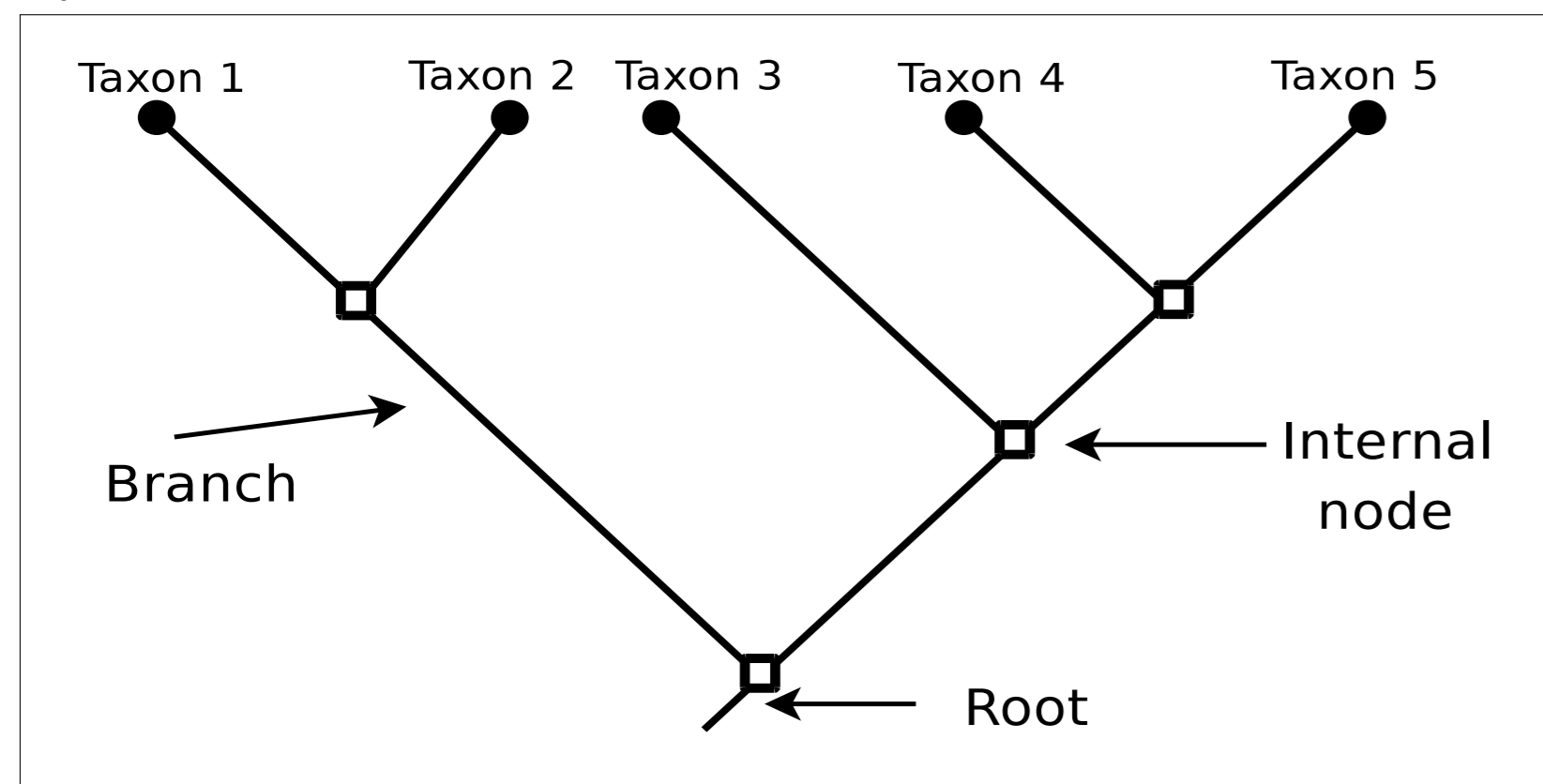
# Parallel numerical methods for inferring Phylogenies

Felipe Fernandes Albrecht Nelson Antonio Borges Garcia  
Instituto Militar de Engenharia

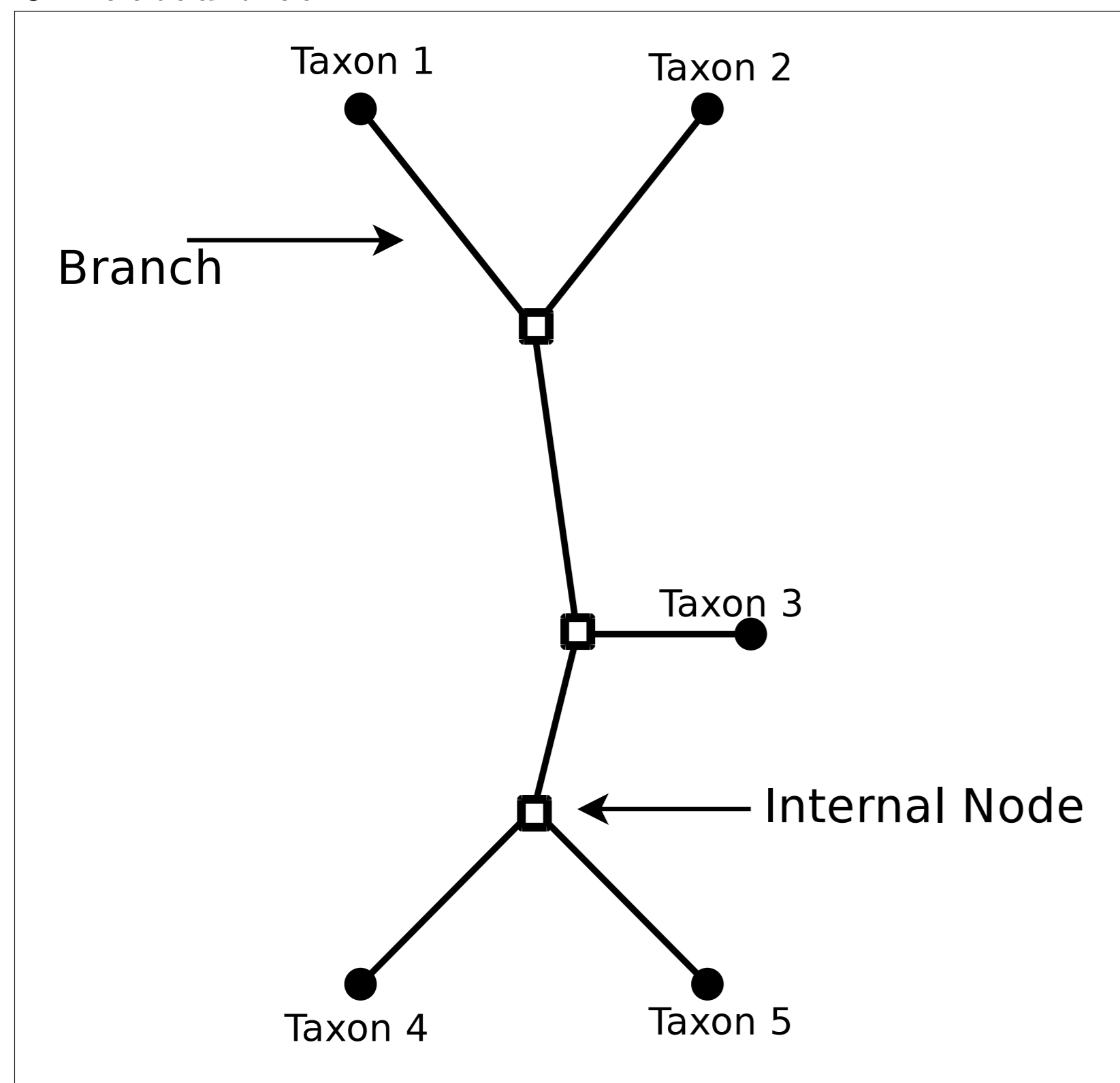
## Introduction

- Molecular phylogeny is used to gain information on an organism's evolutionary relationships.
- The result of a molecular phylogenetic analysis is expressed in a so-called phylogenetic tree.

– Rooted tree:



– Unrooted tree:



- Two main phylogenetics methods:
  - Molecular sequence:
    - \* maximum likelihood
    - \* maximum parsimony methods.
  - Distance matrix:
    - \* Neighbor-Joining
    - \* UPGMA
    - \* Least Squares method

## Motivation

- This work is based on Least Squares method, that is a distance matrix method where an unrooted tree is returned as result.
- This method has an objective function (equation 1) that represents the inferred tree quality and this method

$$Q = \sum_{i=1}^n \sum_{j=1}^n w_{ij} (D_{ij} - d_{ij})^2 \quad (1)$$

- Analyzing the previous work [1], we see that the branches length calculations is the most expensive step.
- This work aims the parallelization of this calculation to speed up the inference process.
- The branches length can be seen as a linear system:

$$Av = b \quad (2)$$

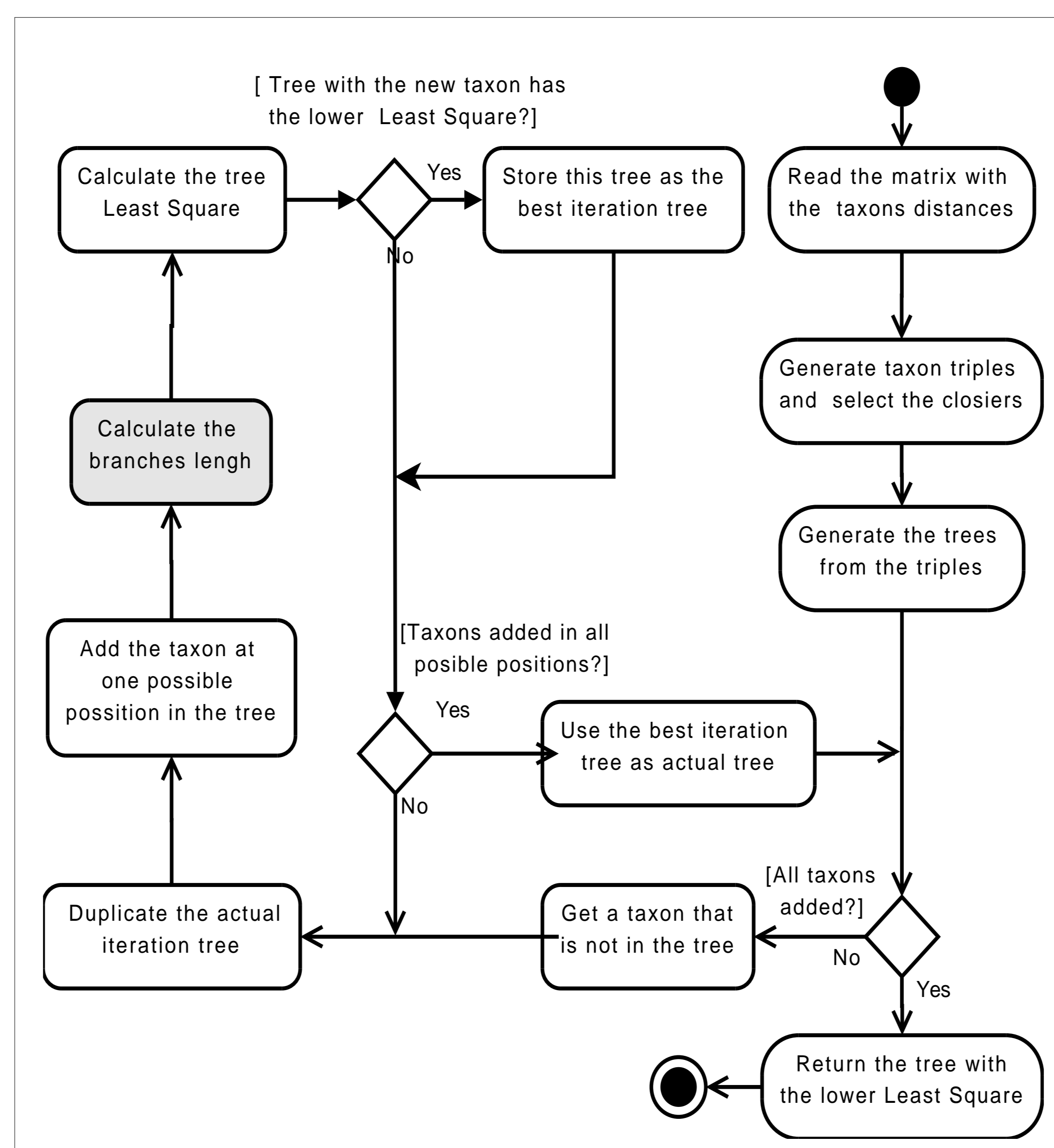
Being  $A$  the transpose topology matrix ( $X$ ) multiplied itself ( $X^t X$ ),  $b = X^t d$  is a vector associated to the independent term. The vector  $d$  represents the observed distances between the taxa given by the distance matrix. The vector  $v$  represents the branches distances between the taxa in the tree that we want to calculate.

## Objectives

- To parallelize the linear system resolution.
- With more machines working, to reduce the execution total time with minor changes in the trees quality.
- Reduce the time processing to solve the numerical linear system when the variables count grow up.
- To study the possibles techniques to parallelize linear systems related with numerical phylogenetics.

## The algorithm

The simplified version of the algorithm is shown bellow.



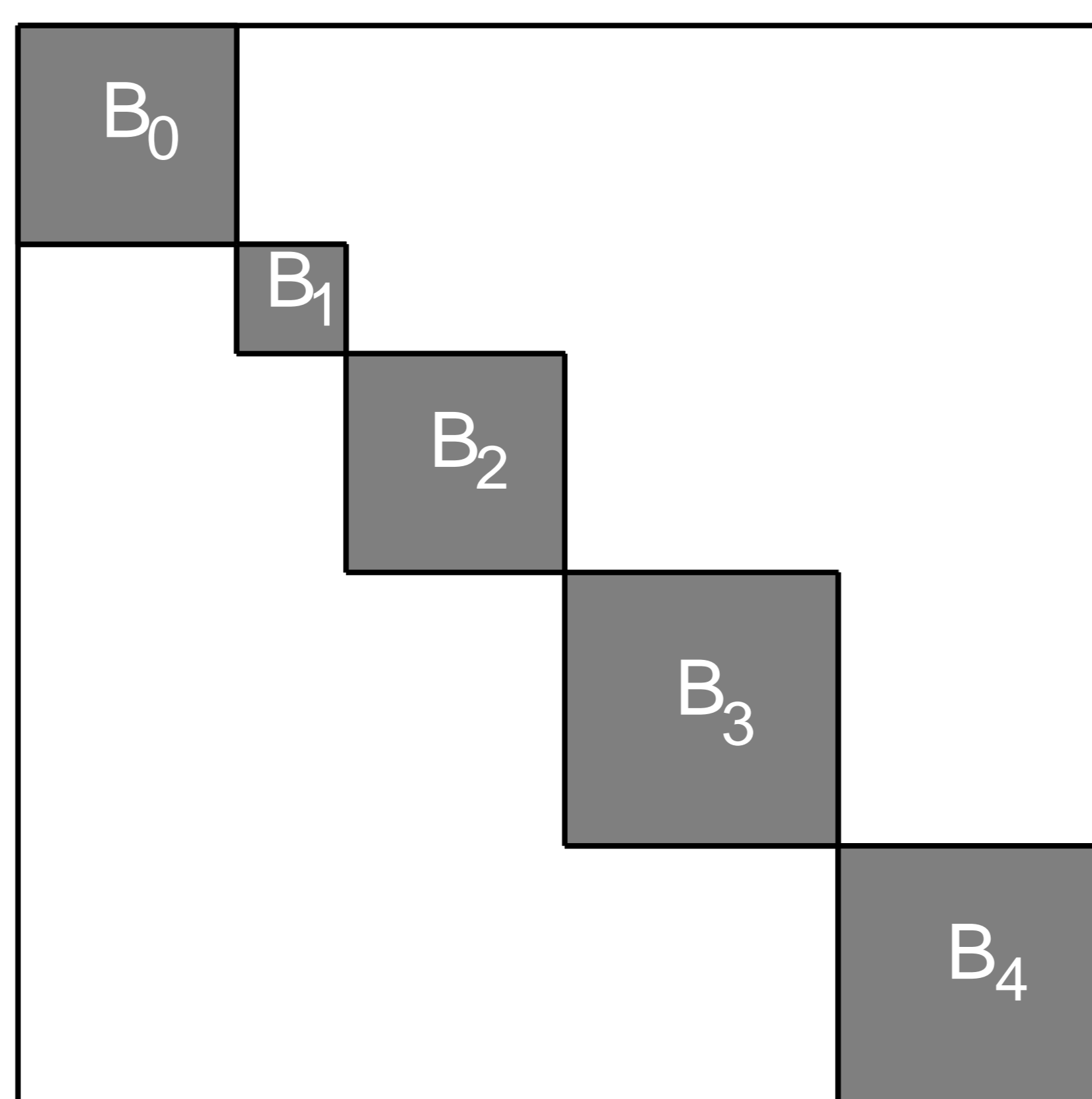
## Studied methods

To solve the linear system  $Av = b$ , three methods was tested:

- Block Jacobi (BJM) method by using  $B_k^{-1}$  in each iteration
- Conjugate gradient method (CGM):  
 $\min J(v) = \frac{1}{2}(Av, v) - (b, v), v \in \mathbb{R}^n$
- Preconditioned conjugate gradient method by block Jacobi (BPCGM):  $\min J(v) = \frac{1}{2}(C^{-1}Av, v) - (C^{-1}b, v), v \in \mathbb{R}^n$

## Partitioned block matrix

The Block Jacobi and Preconditioned conjugate gradient methods use a partitioned block matrix:



## The BPCGM Algorithm

- $r_{k_0} = A_k v_{k_0} - b_k$   
Update  $r_{k_0}$  in all  $p$  processors
- $g_{k_0} = B_k^{-1} r_{k_0}$   
 $p_{k_0} = -g_{k_0}$   
Update  $p_{k_0}$  through all  $p$  processors
- For  $j = 0, 1, \dots, n$ 
  - $\alpha_{k_j} = r_{k_j}^t g_{k_j} / p_{k_j}^t A_k p_{k_j}$   
Reduce  $(\alpha_{k_j})$ ;  $(\alpha_j = \sum \alpha_{k_j})$
  - $v_{k_{j+1}} = v_{k_j} + \alpha_j p_{k_j}$   
If  $\text{abs}(v_{k_{j+1}} - v_{k_j}) < \text{tol}$ , then end.
  - $r_{k_{j+1}} = r_{k_j} + \alpha_j A_k p_{k_j}$   
Update  $r_{k_{j+1}}$  through all  $p$  processors
  - $g_{k_{j+1}} = B_k^{-1} r_{k_{j+1}}$   
Update  $g_{k_{j+1}}$  through all  $p$  processors
  - $\beta_{k_j} = r_{k_{j+1}}^t g_{k_{j+1}} / r_{k_j}^t g_{k_j}$   
Reduce  $(\beta_{k_j})$ ;  $(\beta_j = \sum \beta_{k_j})$
  - $p_{k_{j+1}} = -g_{k_{j+1}} + \beta_j p_{k_j}$   
Update  $p_{k_{j+1}}$  through all  $p$  processors

## Implementation

- The methods are being implemented to obtain results about time and memory consumption.
- The MPI[5] standard with LAM[2] implementation is used for multiprocess communication.
- For multiprocessor, not multicomputer cluster, the methods can be implemented using threads and shared memory.

## Results

The methods was tested with a matrix with 149 rows by 149 columns.

- The inversion method has an growing order of  $O(n^3)$ , doing  $149^3$  operations, resulting at all 3307949 operations.
- The CGM method has also  $O(n^3)$ , but does  $114 \cdot (149^2)$  operations, resulting 2530914 operations, a gain of 30%.
- The BPCGM method has also  $O(n^3)$ , but does  $710 \cdot (149^2)$  operations, resulting 1576271 operations, a gain of 50% in relation of inversion method.
- The BJM converge more slowly, but used as preconditioner of CGM, its accelerate the convergence.

## Conclusions

- This work analyzed some methods to solve the linear system associate with the Least Squares method.
- The BPCGM converge faster then others studied methods.
- The communication costs must be analysed with care, because they can be a bottleneck in the executions in the message passing environment.

## References

- [1] F. F. Albrecht, J. F. Hübner, and A. M. R. Dávila. A distributed algorithm for phylogenetics inference. In BSB 2007 Poster Proceedings, pages 66–69, Angra dos Reis, RJ, 2007. Brazilian Symposium on Bioinformatics, Brazilian Computer Society.
- [2] G. Burns, R. Daoud, and J. Vaigl. Lam. In Proceedings..., pages 379–386, Toronto, 1994. Supercomputing Symposium '94, University of Toronto.
- [3] J. Felsenstein. Phylip (phylogeny inference package) version 3.6, 2005.
- [4] F. S. Foundation. Gcc, the gcc compiler collection, 2006.
- [5] MPI. Message passing interface, 2006.