



V.P.IVANNIKOV MEMORIAL WORKSHOP

# *How To Go Beyond the Limitations of the Top500 Methodology?*

*Prof. Vladimir Voevodin*

*Deputy Director RCC MSU*

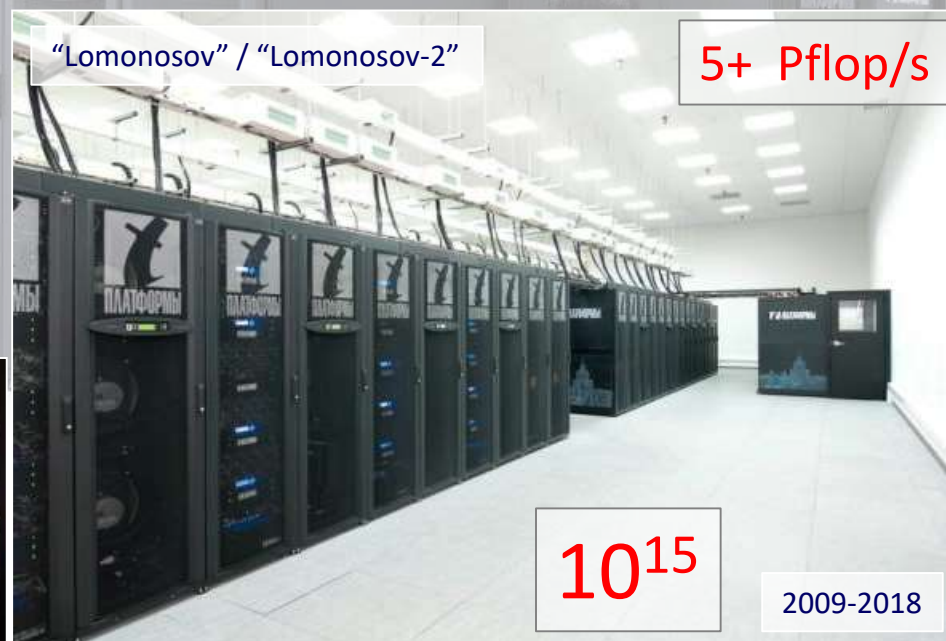
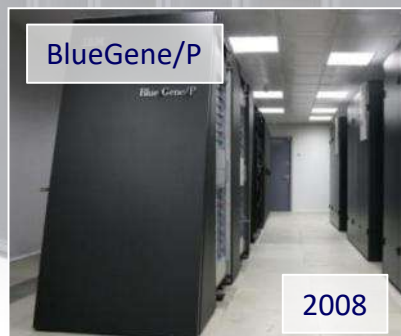
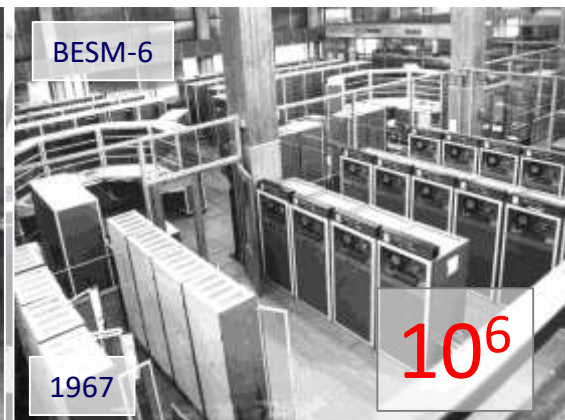
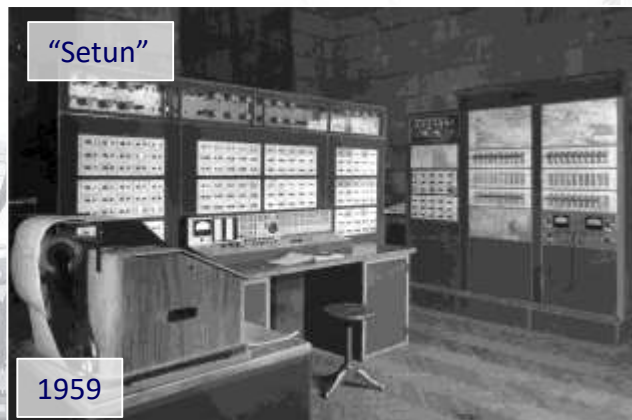
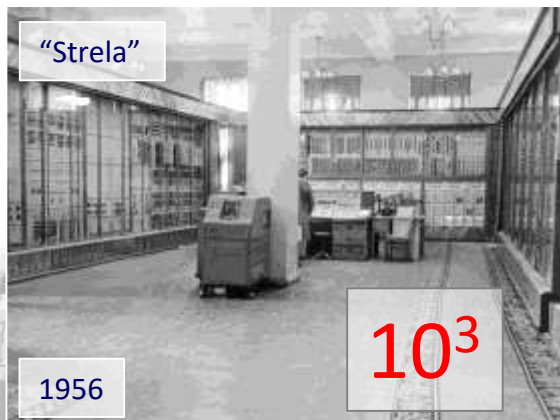
*Head of Department on Supercomputers and Quantum Informatics CMC MSU*

*Corresponding Member of RAS*

*voevodin@parallel.ru*

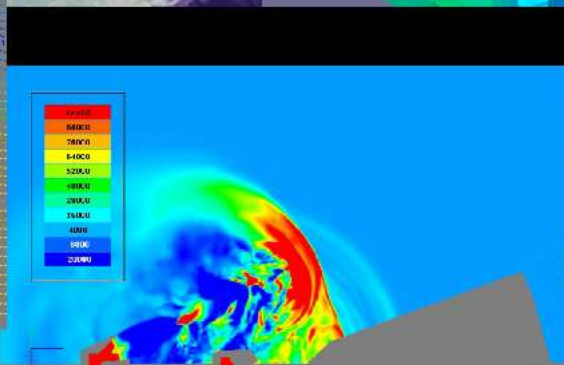
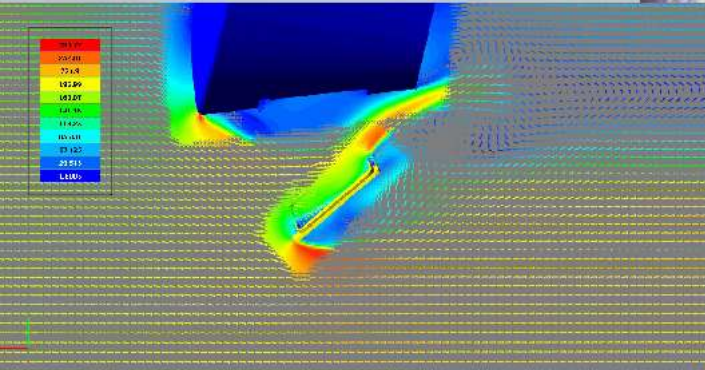
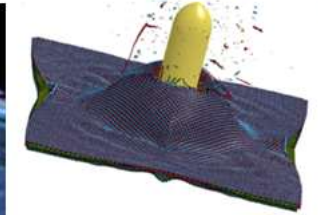
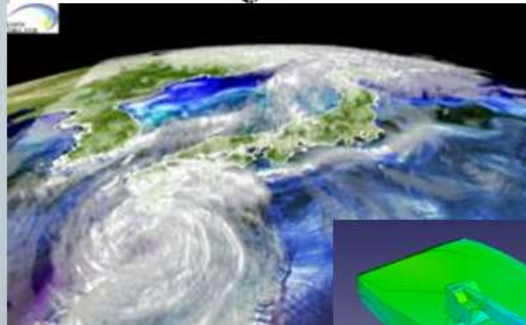
*May, 3<sup>rd</sup> 2018, Erevan, Armenia*

# Computing History of Moscow State University (from 1956 up to now)



# Supercomputer technologies are everywhere...

(how to choose the best computing platform for solving the problem?)



# “Top500” methodology to compare computing platforms (Top500, Graph500, HPCG)

problems for evaluation of computer platforms



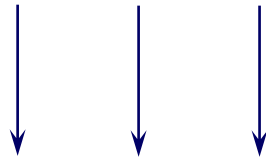
High Performance Linpack  
Benchmark  
Top500.org



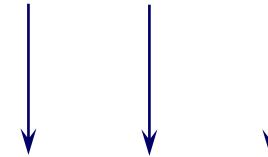
BFS & SSSP Graph  
Benchmarks  
Graph500.org



High Performance Conjugate Gradients  
Benchmark  
hpcg-benchmark.org



Well-known  
theoretical potential



Well-described  
community experience

# “Top500” methodology to compare computing platforms (Top500, Graph500, HPCG)

problems for evaluation of computer platforms



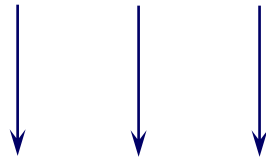
High Performance Linpack  
Benchmark  
Top500.org



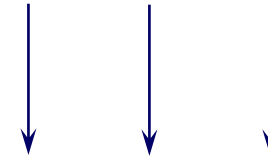
BFS & SSSP Graph  
Benchmarks  
Graph500.org



High Performance Conjugate Gradients  
Benchmark  
hpcg-benchmark.org



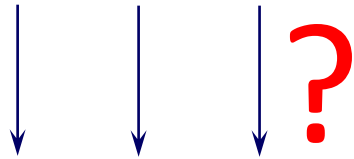
Well-known  
theoretical potential



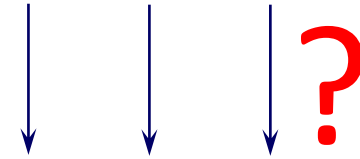
Well-described  
community experience

# *General methodology to compare computing platforms* (using any algorithm)

problems for evaluation of computer platforms

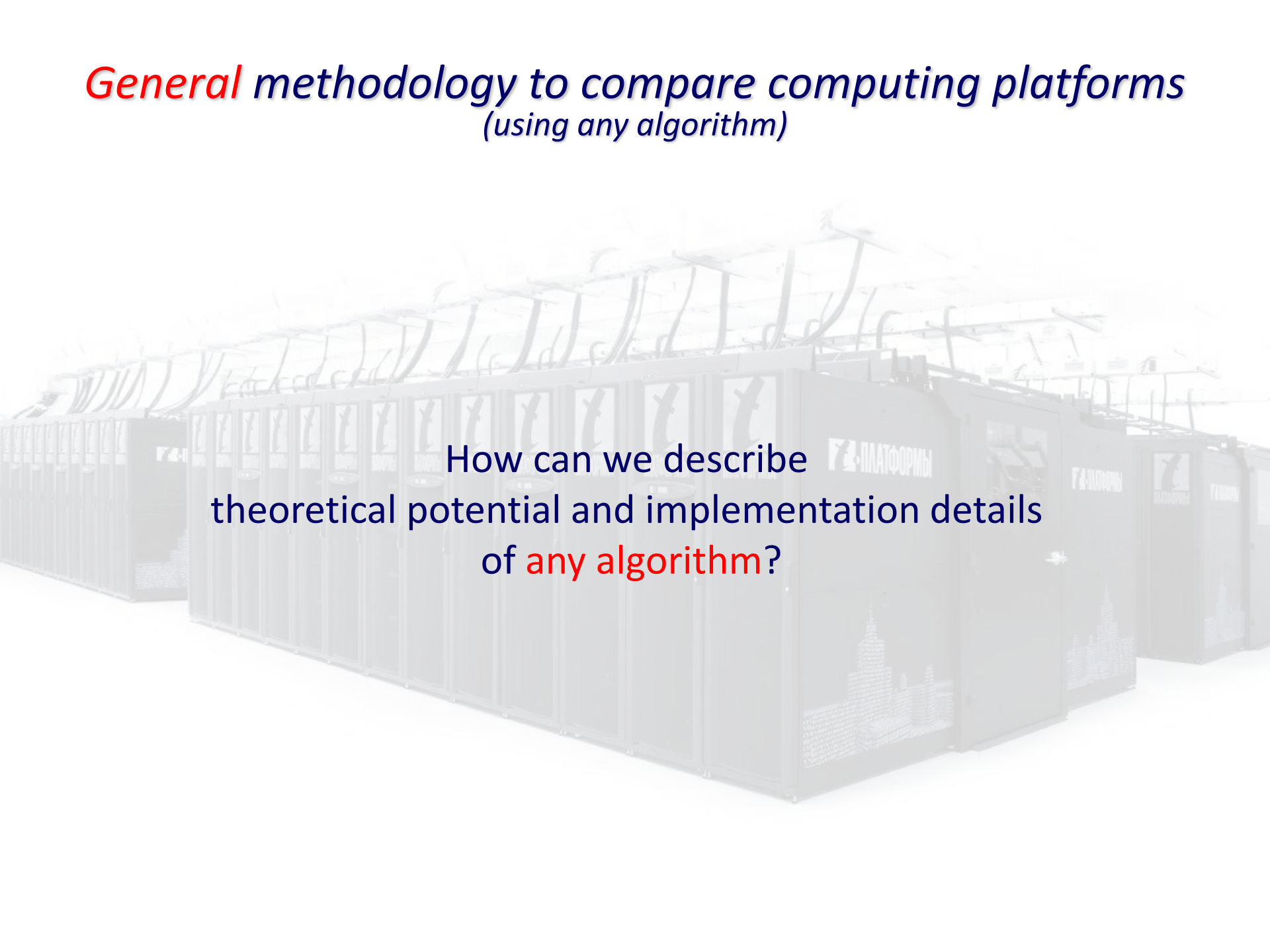


Well-known  
theoretical potential



Well-described  
community experience

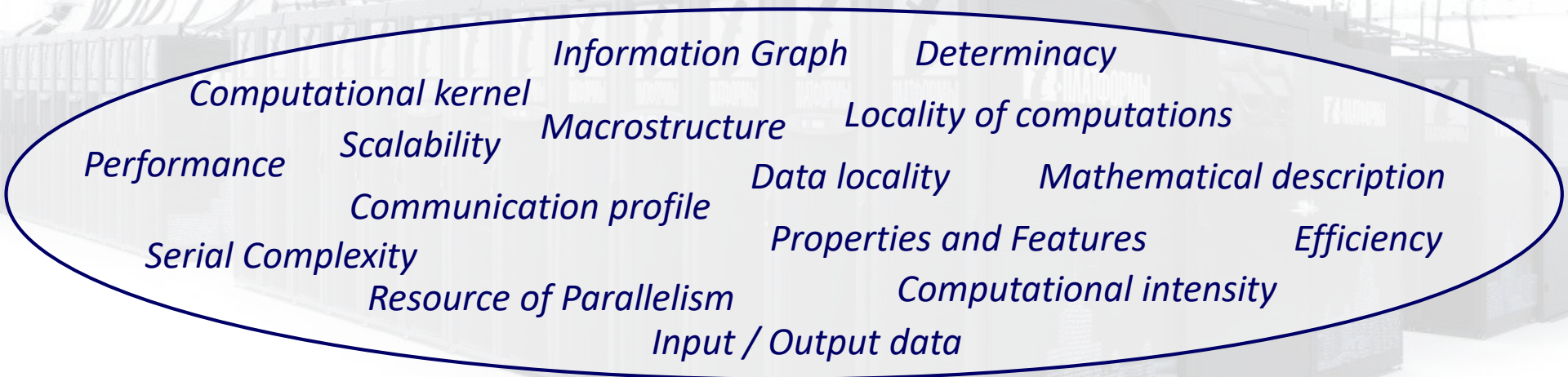
# *General methodology to compare computing platforms* *(using any algorithm)*



How can we describe  
theoretical potential and implementation details  
of **any algorithm**?

# *Description of Algorithms*

*(What features of algorithms should be included in the description?)*



*Information Graph      Determinacy*

*Computational kernel      Locality of computations*

*Performance      Scalability      Macrostructure      Data locality      Mathematical description*

*Communication profile      Properties and Features      Efficiency*

*Serial Complexity      Computational intensity*

*Resource of Parallelism      Input / Output data*



# Description of Algorithms

(What features of algorithms should be included in the description?)

For positive definite Hermitian matrices (*symmetric matrices in the real case*), we use the decomposition  $A = LL^*$ , where  $L$  is the lower triangular matrix. Another form of the Cholesky decomposition is  $A = U^*U$ , where  $U$  is the upper triangular matrix. These forms of the Cholesky decomposition are equivalent in the sense of the amount of arithmetic operations and are different in the sense of data representation.

General Description

The essence of this decomposition consists in the implementation of formulas obtained uniquely for the elements of the matrix  $L$  from

Input data: a symmetric positive definite matrix  $A$  whose elements are denoted by  $a_{ij}$ .

Output data: the lower triangular matrix  $L$  whose elements are denoted by  $l_{ij}$ .

The Cholesky algorithm can be represented in the form

$$l_{11} = \sqrt{a_{11}}, \quad \text{Mathematical Description}$$

$$l_{j1} = \frac{a_{j1}}{l_{11}}, \quad j \in [2, n],$$

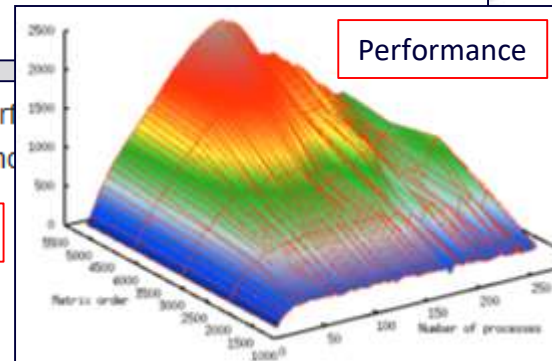
$$l_{ii} = \sqrt{a_{ii} - \sum_{p=1}^{i-1} l_{ip}^2}, \quad i \in [2, n],$$

$$l_{ji} = \left( a_{ji} - \sum_{p=1}^{i-1} l_{ip}l_{jp} \right) / l_{ii}, \quad i \in [2, n-1], j \in [i+1, n].$$

The following number of operations should be performed for the matrix of order  $n$  using a serial version of the Cholesky algorithm:

- $n$  square roots,
- $\frac{n(n-1)}{2}$  divisions,
- $\frac{n^3-n}{6}$  multiplications and  $\frac{n^3-n}{6}$  additions (subtractions): the main amount of computational work.

Serial Complexity



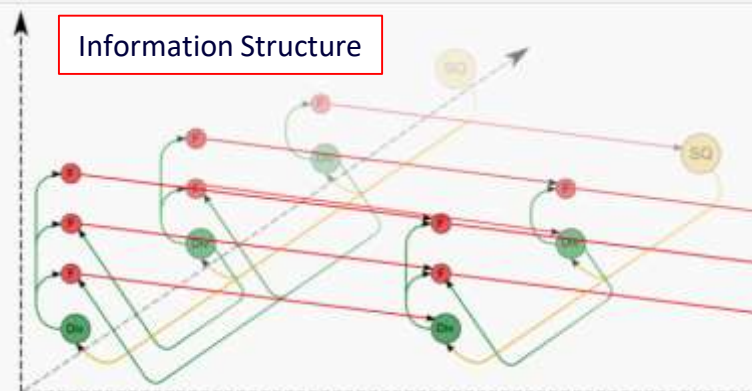
Performance

A computational kernel of its serial version can be composed of  $\frac{n(n-1)}{2}$  dot products

$$\sum_{p=1}^{i-1} l_{ip}l_{jp}$$

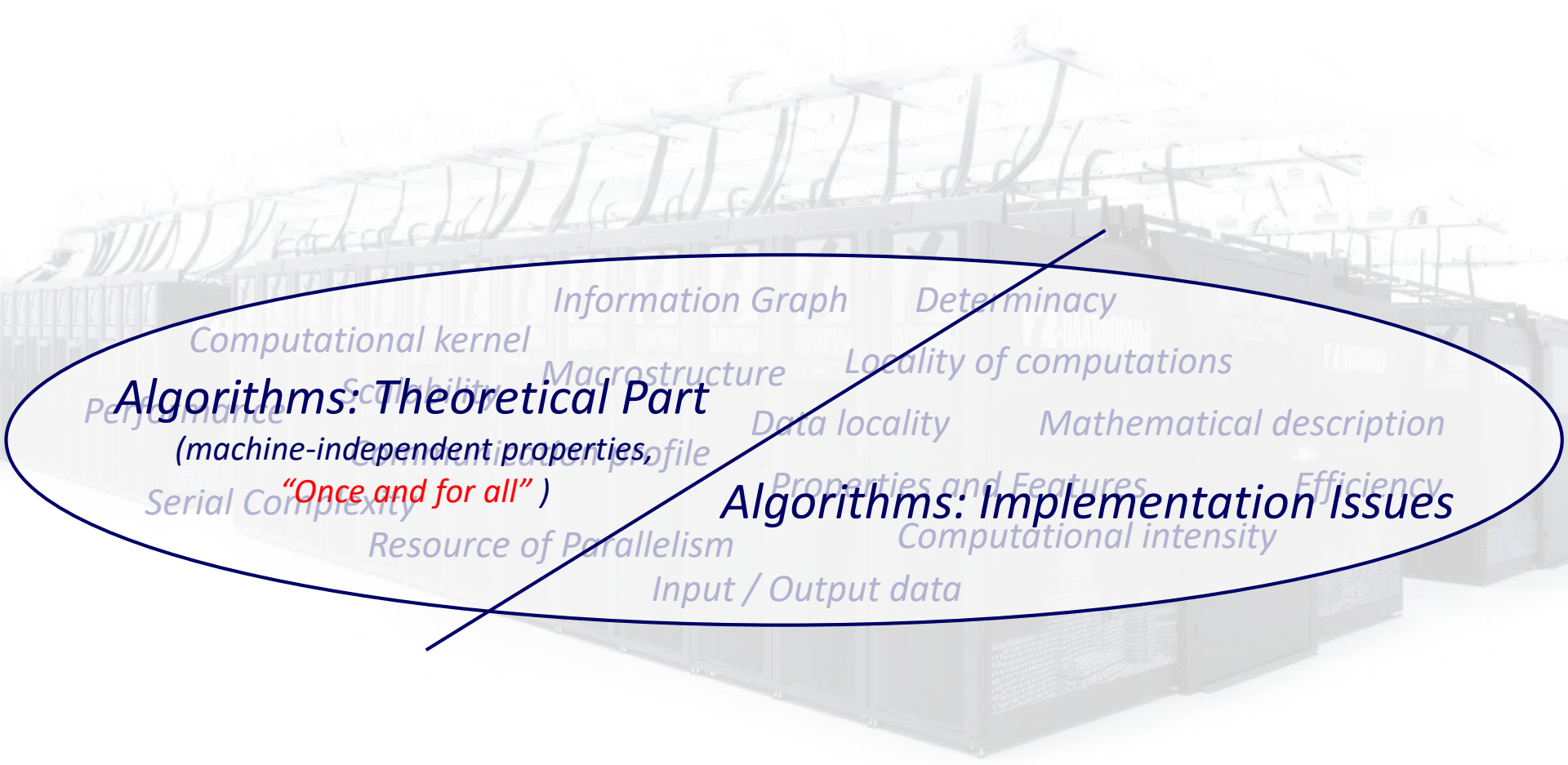
Computational Kernel

Information Structure



# Description of Algorithms

(What features of algorithms should be included in the description?)



## Algorithms: Theoretical Part

(machine-independent properties,  
"Once and for all" )

## Algorithms: Implementation Issues

Information Graph    Determinacy  
Computational kernel    Locality of computations  
Macrostructure    Data locality    Mathematical description  
Serial Complexity    Resource of Parallelism    Computational intensity  
Input / Output data

Файл Правка Вид Журнал Закладки Инструменты Справка

Open Encyclopedia of Par... x +

algowiki-project.org/en/Open\_Encyclopedia\_of\_Parallel\_Algorithmic\_Features

Create account Log in

Page: Discussion Read View source View history Search

## Open Encyclopedia of Parallel Algorithmic Features

Main page  
Forum  
Recent changes

File storage  
New files  
Upload file

Tools  
What links here  
Related changes  
Special pages  
Printable version  
Permanent link  
Page information

In other languages  
Русский

**Welcome! Join us!**

AlgoWiki is an open encyclopedia of **algorithms' properties and features of their implementations** on different hardware and software platforms from mobile to extreme scale, which allows for collaboration with the worldwide computing community on algorithm descriptions.

AlgoWiki provides an exhaustive description of an algorithm. In addition to classical algorithm properties such as serial complexity, AlgoWiki also presents additional information, which together provides a complete description of the algorithm: its parallel complexity, parallel structure, determinacy, data locality, performance and scalability estimates, communication profiles for specific implementations, and many others.

Read more: [About project.](#)

**Project structure**

Algorithm classification — the main section of AlgoWiki which contains descriptions of all algorithms. Algorithms are added to the appropriate category of the classification, and classification is expanded with new sections if necessary.

**Featured article**

### Cholesky decomposition

#### 1 Properties and structure of the algorithm

##### 1.1 General description

The **Cholesky decomposition algorithm** was first proposed by Andre-Louis Cholesky (October 15, 1875 - August 31, 1918) at the end of the First World War shortly before he was killed in battle. He was a French military officer and mathematician. The idea of this algorithm was published in 1924 by his fellow officer and, later, was used by Banachiewicz in 1938 [7]. In the Russian mathematical literature, the Cholesky decomposition is also known as the square-root method [1-3] due to the square root operations used in this decomposition and not used in Gaussian elimination.

Originally, the Cholesky decomposition was used only for dense real symmetric positive definite matrices. At present, the application of this decomposition is much wider. For example, it can also be employed for the case of Hermitian matrices. In order to increase the computing performance, its block versions are often applied.

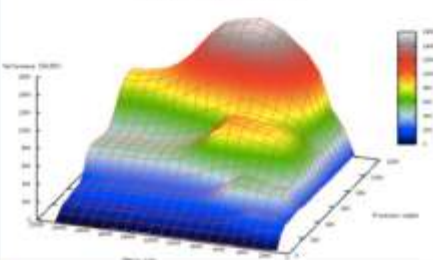
In the case of sparse matrices, the Cholesky decomposition is also widely used as the main stage of a direct method for solving linear systems. In

**Properties of the algorithm:**

- Sequential complexity:  $O(n^3)$
- Height of the parallel form:  $O(n)$
- Width of the parallel form:  $O(n^2)$
- Amount of input data:  $\frac{n(n+1)}{2}$
- Amount of output data:  $\frac{n(n+1)}{2}$

**Today's featured picture**

Matrix multiplication performance



Performance for dense matrix multiplication

**Work organization**

Description of algorithm properties and structure  
[Guides to writing sections of the algorithm's description](#)  
 Glossary  
 Help with editing

**Readiness of articles**

**Finished articles:**

- Single-qubit transform of a state vector
- Two-sided Thomas algorithm, pointwise version
- Poisson equation, solving with DFT
- Thomas algorithm, pointwise version
- Backward substitution

# AlgoWiki: algorithm classification (tree structure)

## Algorithm classification

### 1 Linear algebra problems

#### 1.1 Matrix and vector operations

##### 1.1.1 Vector operations

- 1. **Dot product**
  1. **Dot product**
  2. **Parallel prefix scan algorithm using parallel summation**
- 2. **Uniform norm of a vector** (Fast version, exact/parallel version)
- 3. **Dependency**
- 4. **The serial-parallel summation method**

##### 1.1.2 Matrix-vector operations

###### 1.1.2.1 Multiplying a nonnegative matrix by a vector

- 1. **Dense matrix-vector multiplication**

###### 1.1.2.2 Multiplying a matrix of special form by a vector

- 1. **Fourier transform**
  1. **Fast Fourier transform for complex dimension**
    - 1. **Fast Fourier transform for powers of two**
    - 2. **Cooley-Tukey Fast Fourier Transform (radix-2 case)**
    - 3. **Fast Fourier transform for even powers of two**
    - 4. **Fast Fourier transform for complex dimension with empty prime divisors (2,3,5,7)**
    - 5. **Fast Fourier transform for prime dimension**

##### 1.1.3 Matrix operations

- 1. **Dense matrix multiplication**
  1. **Dense matrix multiplication (serial version for real matrices)**
  2. **Strassen's algorithm**

#### 1.2 Matrix decompositions

##### 1.2.1 Matrix decomposition problem

###### 1.2.1.1 Triangular decompositions

- 1. **LU decomposition (finding the LU decomposition)**
  1. **Gaussian elimination using Gaussian elimination without pivoting**
  2. **LU decomposition via Gaussian elimination**
  3. **Gaussian elimination, compare schemes for triangular matrices and its modifications**
    1. **Compare schemes for Gaussian elimination: Dense matrix**
    2. **Compare schemes for Gaussian elimination: Triangular matrix**
      - 1. **Gaussian elimination, compare schemes for triangular matrices, serial version**
      - 2. **Serial doubling algorithm for the LU decomposition of a triangular matrix**
      - 3. **Serial-parallel algorithm for the LU decomposition of a triangular matrix**

###### 1.2.1.2 LU decomposition using Gaussian elimination with pivoting

- 1. **Gaussian elimination with column pivoting**
- 2. **Gaussian elimination with row pivoting**
- 3. **Gaussian elimination with diagonal pivoting**
- 4. **Gaussian elimination with complete pivoting**

###### 1.2.1.3 Cholesky method

- 1. **Cholesky decomposition**

##### 1.2.2 Available triangular decompositions for matrices of special form

###### 1.2.2.1 Unitary-orthogonal factorizations

- 1. **QR decomposition of dense nonsingular matrices**
  1. **Givens (rotations) method for the QR decomposition of a matrix**
    - 1. **Givens method**
  2. **Householder (reflections) method for the QR decomposition of a matrix**
  3. **Householder (reflections) method for the QR decomposition of a sparse matrix (real/complex version)**
  4. **Orthogonalization method**
    1. **Classical orthogonalization method**
    2. **Orthogonalization method with reorthogonalization**
  5. **Triangular decomposition of Gram matrix**
- 2. **QR decomposition methods for dense Hessenberg matrices**
  1. **Givens (rotations) method for the QR decomposition of a (real) Hessenberg matrix**
  2. **Householder (reflections) method for the QR decomposition of a (real) Hessenberg matrix**

###### 1.2.2.2 Reducing matrices to complex forms

- 1. **Unitary reductions to Hessenberg form**
  1. **Householder (reflections) method for reducing a matrix to Hessenberg form**
  2. **Classical Givens (rotations) method for reducing a matrix to Hessenberg form**
  3. **Classical Givens (rotations) method for reducing a matrix to Hessenberg form**
- 2. **Unitary reductions to bidiagonal form**
  1. **Householder (reflections) method for reducing a symmetric matrix to bidiagonal form**
  2. **Householder (reflections) method for reducing a complex Hermitian matrix to a symmetric tridiagonal form**
- 3. **Givens (rotations) reduction to bidiagonal form**

###### 1.2.2.3 Eigenvalue decomposition (finding eigenvalues and eigenvectors)

- 1. **Unitary reductions to Schur form**
  1. **Householder (reflections) reduction of a matrix to bidiagonal form**
  2. **Givens (rotations) reduction of a matrix to bidiagonal form**

###### 1.2.2.4 Singular value decomposition

- 1. **Singular value decomposition (finding singular values and singular vectors)**

### 1.3 Solving systems of linear algebraic equations

#### 1.3.1 Direct methods

##### 1.3.1.1 Unpack benchmark

##### 1.3.1.2 Libraries of a special form

##### 1.3.1.3 Triangular matrices

- 1. **Forward substitution**
- 2. **Backward substitution**
- 3. **Singular matrices**
  1. **Forward and backward substitution for bidiagonal matrices**
  2. **Serial doubling algorithm for solving bidiagonal SLSs**
  3. **Serial-parallel variant of the backward substitution**
  4. **Serial-parallel variant of the backward substitution**

##### 1.3.1.4 Methods for solving tridiagonal SLSs

- 1. **Methods based on the conventional LU decomposition**
  1. **Thomas algorithm**
    1. **Thomas algorithm, complex version**
    2. **Revised Thomas algorithm, complex version**
  2. **Serial doubling algorithm**
    1. **Serial doubling algorithm for the LU decomposition of tridiagonal matrices**
    2. **Serial doubling algorithm for solving tridiagonal SLSs**
  3. **Serial-parallel method for solving tridiagonal matrices based on the LU decomposition and backward substitutions**

##### 1.3.1.5 Other methods

- 1. **Reduction method**
  1. **Complex reduction method**
  2. **Reduction method repeated for a new right-hand side**
- 2. **Two-sided Thomas algorithm**
  1. **Two-sided Thomas algorithm, complex version**
  2. **Revised two-sided Thomas algorithm, complex version**
- 3. **Cyclic reduction**
  1. **Complex cyclic reduction**
  2. **Cyclic reduction repeated for a new right-hand side**
- 4. **Block bordering method**

##### 1.3.1.6 Methods for solving block triangular matrices

- 1. **Block forward substitution (real version)**
- 2. **Block backward substitution (real version)**
- 3. **Methods for solving block bidiagonal matrices**
  1. **Forward and backward substitution for block bidiagonal matrices**
  2. **Serial doubling algorithm for solving block bidiagonal matrices**

##### 1.3.1.7 Methods for solving block bidiagonal matrices

- 1. **Methods based on the conventional LU decomposition**
  1. **Block Thomas algorithm**
  2. **Serial-parallel method for solving block systems of linear algebraic equations based on the LU decomposition and backward substitutions**
- 2. **Other methods**
  1. **Two-sided Thomas algorithm, block version**
  2. **Block cyclic reduction**
  3. **Block bordering method**

##### 1.3.1.8 Solving systems of linear algebraic equations with coefficient matrices of special form whose inverses are known

- 2. **Serial methods for solving systems of linear algebraic equations**
  1. **High Performance Conjugate Gradient (HPCG) benchmark**
  2. **Block-cyclic preconditioned method (BCGSs)**
  3. **Block-cyclic algorithm**

#### 1.4 Solving eigenvalue problems

##### 1.4.1 Eigenvalue decomposition (finding eigenvalues and eigenvectors)

###### 1.4.1.1 QR algorithm

- 1. **QR algorithm as implemented in SCLUPACK**
  1. **Classical Givens (rotations) method for reducing a matrix to Hessenberg form**
  2. **Hessenberg QR algorithm as implemented in SCLUPACK**
- 2. **Symmetric QR algorithm as implemented in SCLUPACK**
  1. **Householder (reflections) method for reducing a symmetric matrix to bidiagonal form**
  2. **Symmetric tridiagonal QR algorithm as implemented in SCLUPACK**
  3. **QR algorithm for complex Hermitian matrices as implemented in SCLUPACK**
- 3. **Symmetric tridiagonal QR algorithm for reducing a complex Hermitian matrix to a symmetric tridiagonal form**
- 4. **Symmetric tridiagonal QR algorithm as implemented in SCLUPACK**

###### 1.4.1.2 The Jacobi (rotations) method for solving the symmetric eigenvalue problem

- 1. **The classical Jacobi (rotations) method with pivoting for symmetric matrices**
  1. **Serial Jacobi (rotations) method for symmetric matrices**
  2. **Serial Jacobi (rotations) method with threshold for symmetric matrices**
- 2. **Parlett's algorithm**
- 3. **Lanczos algorithm**

###### 1.4.1.3 Parlett's algorithm in an eigenproblem (with reorthogonalization)

- 2. **Parlett's algorithm in an eigenproblem (with reorthogonalization)**
  1. **Method of deflation**
  2. **Singular value decomposition (finding singular values and singular vectors)**
    1. **Jacobi (rotations) method for finding singular values**
    2. **Serial Jacobi (rotations) method for finding singular values**
    3. **Jacobi method with a specific choice of rotator for finding singular values**

- 3. **QR algorithm as applied to singular value decomposition arrays**

##### 1.4.2 Algorithms of polynomial

- 1. **Horner's method**

### 2 Algorithms on lists and arrays

#### 2.1 Search algorithms

- 1. **Linear search: Finding a term in an arbitrary list:  $O(n)$**
- 2. **Binary search: Finding the position of a target value within a sorted array:  $O(\log(n))$**

#### 2.2 Sorting algorithms

- 1. **Binary tree sort**
- 2. **Bubble sort**
- 3. **Large constant and parallel variants**

#### 2.3 Graph algorithms

- 1. **Graph traversal**
  1. **Breadth-First search (BFS)**
  2. **Depth-First search (DFS)**
- 2. **Single Source Shortest Path (SSSP)**
  1. **Breadth-First search (BFS) for unweighted graphs**
  2. **Dijkstra's algorithm**
  3. **Bellman-Ford algorithm**
  4. **Shortest path algorithm**
- 3. **All Pairs Shortest Path (APSP)**
  1. **Floyd-Warshall algorithm**
  2. **Floyd-Cycle-Finding algorithm**
  3. **Transitive closure of a directed graph**
- 4. **Longest path algorithm**
- 5. **Construction of the minimum spanning tree (MST)**
  1. **Kruskal's algorithm**
  2. **Prim's algorithm**
  3. **Greedy algorithm**
- 6. **Search for Isomorphic Subgraphs**
  1. **Vitman's algorithm**
  2. **PT algorithm**
- 7. **Graph connectivity**
  1. **Tarjan-Liskovits algorithm for finding the connected components**
  2. **DFS algorithm**
  3. **Tarjan's strongly connected components algorithm**
  4. **DFS algorithm for finding the strongly connected components**
  5. **Tarjan's biconnected components algorithm**
  6. **Tarjan-Liskovits biconnected components algorithm**
  7. **Tarjan's algorithm for finding the bridges of a graph**
  8. **Flow connectivity of a graph**
  9. **Edmond's edge connectivity algorithm**
- 8. **Finding maximal flow in a transportation network**
  1. **Ford-Fulkerson algorithm**
  2. **Edmond-Karp algorithm**
- 9. **Flow network algorithm**
- 10. **Flow network algorithm in a transportation network**
- 11. **Assignment problem**
  1. **Hungarian algorithm**
  2. **Auction algorithm**
  3. **Successive algorithm**
- 12. **Maximal flow centrality algorithm**

### 3 Computational geometry

- 1. **Finding the diameter of a polygon**
- 2. **Finding the convex hull of a polygon**
- 3. **Delaunay triangulation**
- 4. **Traveling salesman**
- 5. **Point-in-polygon problem**
- 6. **Convex polygon intersection**
- 7. **Star-shaped polygon intersection**

#### 3.1 Computer graphics

- 1. **Line drawing algorithms: approximating a line segment discrete graphical media**
- 2. **Drawing visible parts of a three-dimensional scene**
- 3. **Fast moving thresholding methods: image**
- 4. **Global illumination: Ray-tracing of illumination and reflection from other objects**

### 4 Computer analysis and modeling

#### 4.1 Computer benchmarks

- 1. **High Performance Conjugate Gradient (HPCG) benchmark**
- 2. **Unpack benchmark**

#### 4.2 Algorithms of quantum system simulation

- 1. **Algorithms of quantum computation simulation**
  1. **Single-qubit algorithm of a state vector**
  2. **Two-qubit algorithm of a state vector**
  3. **Quantum Fourier transform simulation**

# *AlgoWiki: Problem – Method – Algorithm – Implementation*

*(What do we have for each algorithm in AlgoWiki?)*

*An exhaustive description of the chain)*



Problem



Method



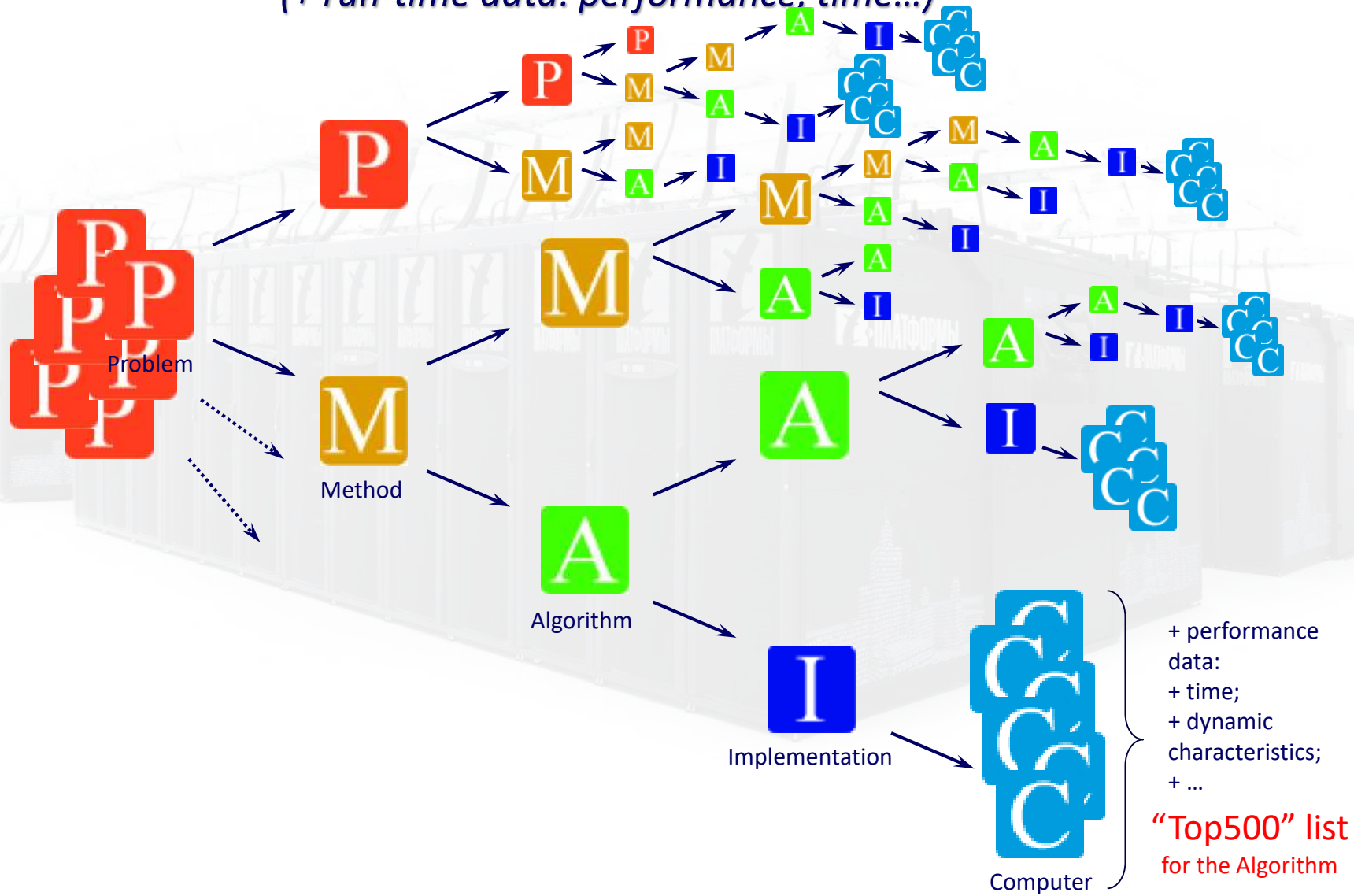
Algorithm



Implementation

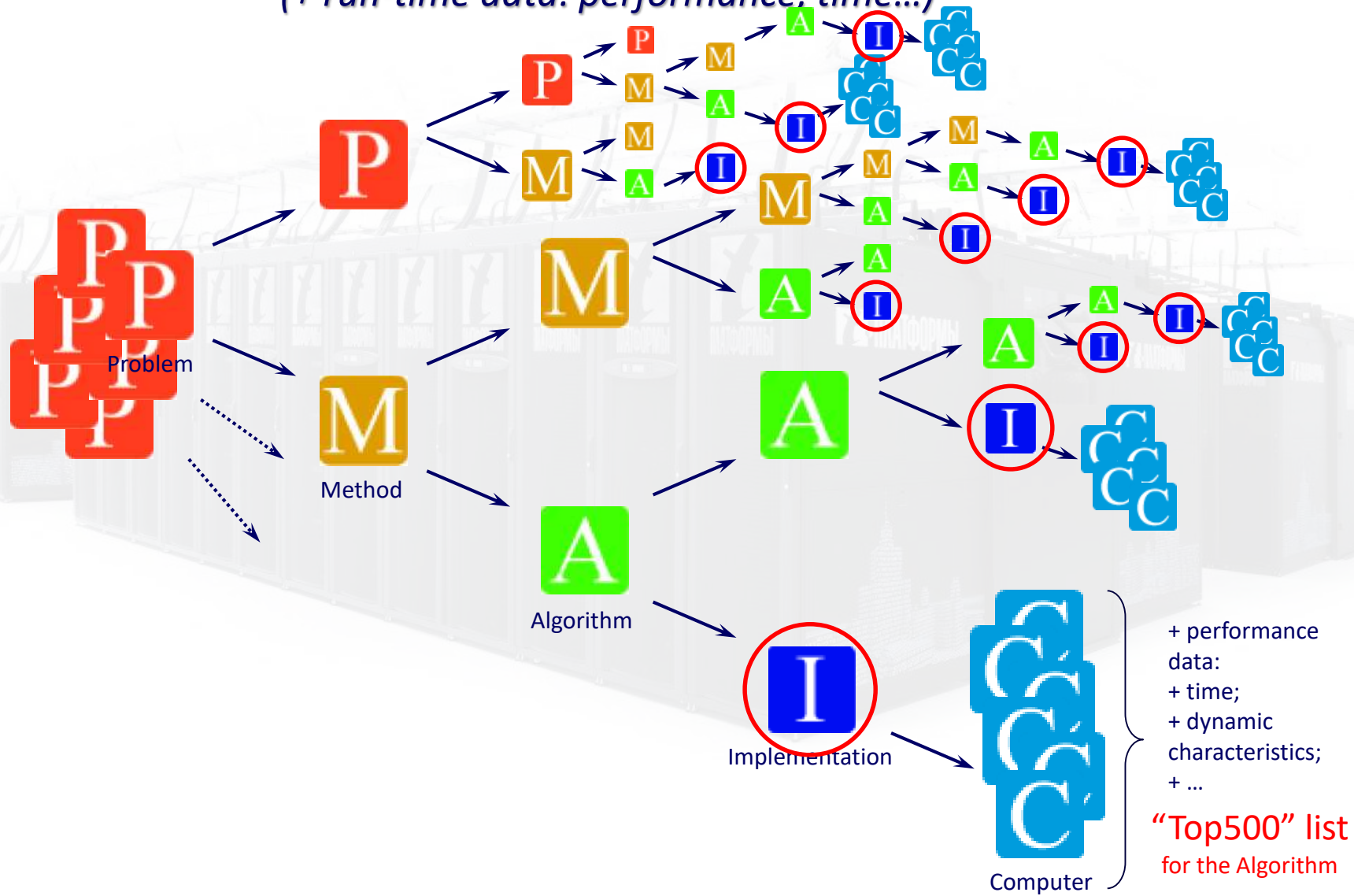
# AlgoWiki: from Problems to various Implementations

(+ run-time data: performance, time...)



# AlgoWiki: from Problems to various Implementations

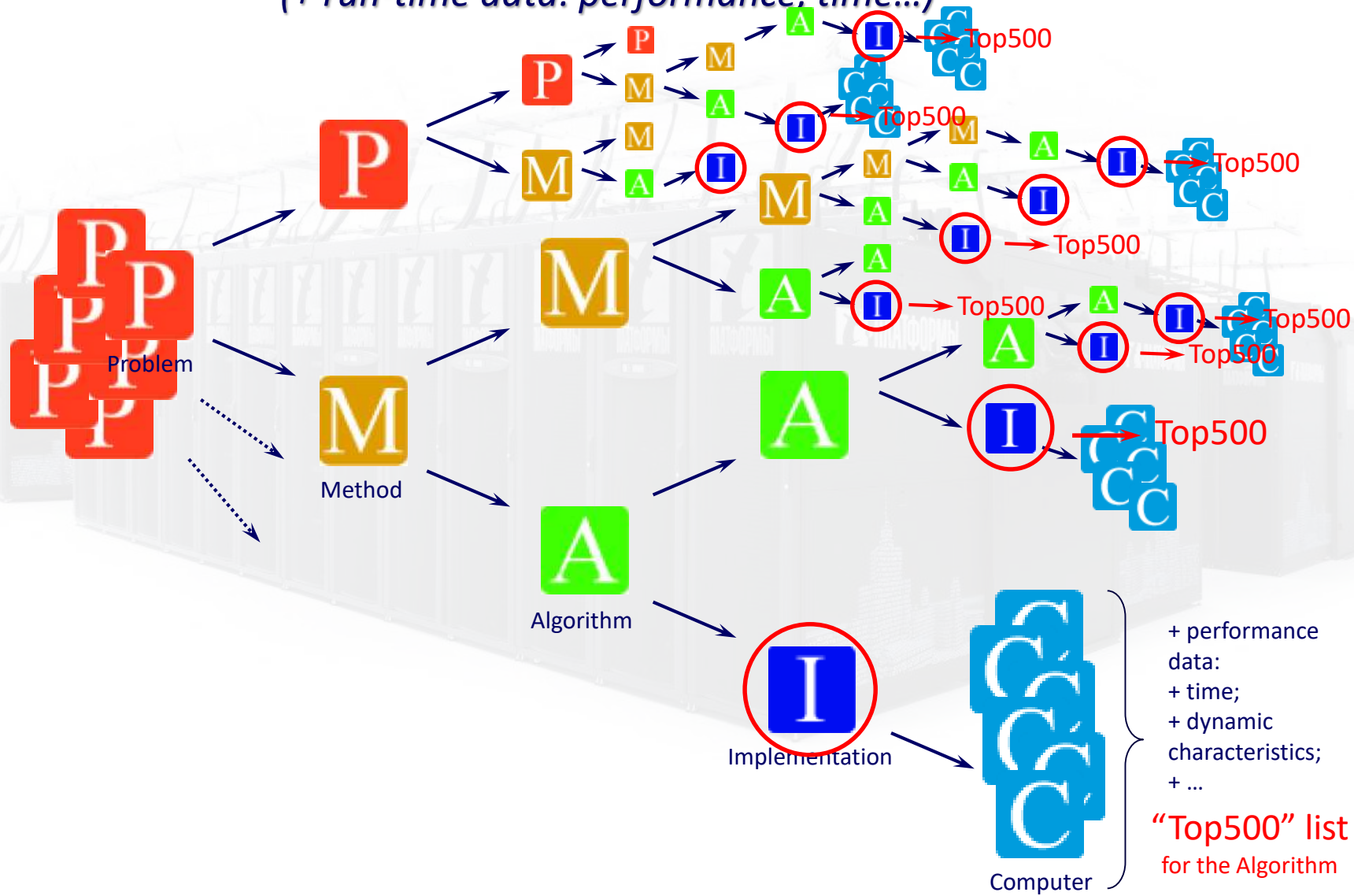
(+ run-time data: performance, time...)



+ performance data;  
+ time;  
+ dynamic characteristics;  
+ ...  
"Top500" list  
for the Algorithm

# AlgoWiki: from Problems to various Implementations

(+ run-time data: performance, time...)





# Algorithm classification and comparison of computers

(what have we had up to now? Three points on the entire set)

## Algorithm classification

### 1 Linear algebra problems

#### 1.1 Matrix and vector operations

##### 1.1.1 Vector operations

1. Dot product summation
  1. Parallel prefix scan algorithm using parallel summation
  2. Uniform norm of a vector (Fast Fourier Transform, serial/parallel version)
  3. Dependency
  4. The serial-parallel summation method

##### 1.1.2 Matrix-vector operations

##### 1.1.2.1 Multiplying a nontrigonal matrix by a vector

1. Dense matrix-vector multiplication

##### 1.1.2.2 Multiplying a matrix of special form by a vector

1. Fourier transform
  1. Fast Fourier transform for complex dimension
    1. Fast Fourier transform for powers-of-two
    2. Cooley-Tukey Fast Fourier Transform (radix-2 case)
    3. Fast Fourier transform for even (complex)
      1. Fast Fourier transform for complex dimension with empty prime divisors (2,3,5,7)
      2. Fast Fourier transform for prime dimension

##### 1.1.3 Matrix operations

1. Dense matrix multiplication
  1. Dense matrix multiplication (serial version for real matrices)
  2. Strassen's algorithm

#### 1.2 Matrix decompositions

##### 1.2.1 Matrix decomposition problem

##### 1.2.1.1 Triangular decompositions

1. LU decomposition (finding the LU decomposition)
  1. LU decomposition using Gaussian elimination (with/without pivoting)
    1. LU decomposition via Gaussian elimination
    2. Gaussian elimination, compact scheme for trigonal matrices and its modifications
      1. Compact scheme for Gaussian elimination: Dense matrix
        1. Gaussian elimination, compact scheme for trigonal matrices, serial version
        2. Serial doubling algorithm for the LU decomposition of a trigonal matrix
        3. Serial-parallel algorithm for the LU decomposition of a trigonal matrix
  2. LU decomposition using Gaussian elimination with pivoting
    1. Gaussian elimination with column pivoting
    2. Gaussian elimination with row pivoting
    3. Gaussian elimination with diagonal pivoting
    4. Gaussian elimination with complete pivoting

Top500

##### 1.2.1.2 Cholesky method

1. Cholesky decomposition

##### 1.2.1.3 Available triangular decompositions for matrices of special form

##### 1.2.1.3.1 Unitary-orthogonal factorizations

1. QR decomposition of dense nonsingular matrices
  1. Gram-Schmidt method for the QR decomposition of a matrix
    1. Householder (reflections) method for the QR decomposition of a matrix
    2. Orthogonalization method
      1. Classical orthogonalization method
      2. Orthogonalization method with Householder
    3. Triangular decomposition of a Gram matrix
  2. QR decomposition methods for dense Hessenberg matrices
    1. Gram-Schmidt method for the QR decomposition of a (real) Hessenberg matrix
    2. Householder (reflections) method for the QR decomposition of a (real) Hessenberg matrix

##### 1.2.1.3.2 Reducing matrices to compact forms

1. Unitary reductions to Hessenberg form
  1. Householder (reflections) method for reducing a matrix to Hessenberg form
    1. Classical (orthogonal) Householder (reflections) method for reducing a matrix to Hessenberg form
    2. Classical (orthogonal) Householder (reflections) method for reducing a matrix to Hessenberg form
  2. Unitary reductions to orthogonal form
    1. Householder (reflections) method for reducing a symmetric matrix to orthogonal form
    2. Gram-Schmidt (reflections) method for reducing a complex Hermitian matrix to a symmetric orthogonal form
  3. Gram-Schmidt (reflections) method for reducing a matrix to orthogonal form

##### 1.2.1.3.3 Singular value decompositions

1. Singular value decomposition (finding eigenvalues and eigenvectors)

### 1.3 Solving systems of linear algebraic equations

#### 1.3.1 Direct methods

1. Unpack benchmark
  1. Triangular matrices
    1. Forward substitution
    2. Backward substitution
    3. Singular matrices
      1. Forward and backward substitution for bidiagonal matrices
      2. Serial doubling algorithm for solving bidiagonal SLSs
      3. Serial-parallel variants for the backward substitution
      4. Serial-parallel variants for solving bidiagonal SLSs

#### 1.3.2 Iterative methods

1. Methods based on the conventional LU decomposition
  1. Thomas algorithm
    1. Thomas algorithm, polynomial version
    2. Reversed Thomas algorithm, polynomial version
  2. Serial doubling algorithm
    1. Serial doubling algorithm for the LU decomposition of trigonal matrices
    2. Serial doubling algorithm for solving bidiagonal SLSs
    3. Serial-parallel variants for solving trigonal matrices based on the LU decomposition and backward substitutions
2. Other methods
  1. Reduction method
    1. Complex reduction method
    2. Reduction method repeated for a new (right-hand side)
  2. The serial Thomas algorithm
    1. The serial Thomas algorithm, polynomial version
    2. Reversed serial Thomas algorithm, polynomial version
  3. Cyclic reduction
    1. Complex cyclic reduction
    2. Cyclic reduction repeated for a new (right-hand side)
  4. Bandwidth method

#### 1.3.3 Iterative methods

1. Methods for solving block triangular matrices
  1. Forward and backward substitution for block bidiagonal matrices
  2. Serial doubling algorithm for solving block bidiagonal matrices
2. Methods for solving block bidiagonal matrices
  1. Forward and backward substitution for block bidiagonal matrices
  2. Serial doubling algorithm for solving block bidiagonal matrices
3. Methods for solving block systems of linear algebraic equations based on the LU decomposition and backward substitutions
  1. Block Thomas algorithm
  2. Serial-parallel method for solving block systems of linear algebraic equations based on the LU decomposition and backward substitutions
4. Other methods
  1. The serial Thomas algorithm, block variant
  2. Block cyclic reduction
  3. Block banding method

### 1.4 Solving eigenvalue problems

#### 1.4.1 Eigenvalue decomposition (finding eigenvalues and eigenvectors)

1. QR algorithm
  1. QR algorithm as implemented in SCLUPACK
    1. Classical (orthogonal) Householder (reflections) method for reducing a matrix to Hessenberg form
    2. Hessenberg QR algorithm as implemented in SCLUPACK
  2. Symmetric QR algorithm as implemented in SCLUPACK
    1. Householder (reflections) method for reducing a symmetric matrix to orthogonal form
    2. Symmetric orthogonal QR algorithm as implemented in SCLUPACK
  3. QR algorithm for complex Hermitian matrices as implemented in SCLUPACK
    1. Householder (reflections) method for reducing a complex Hermitian matrix to a symmetric orthogonal form
    2. Symmetric orthogonal QR algorithm as implemented in SCLUPACK
2. The Jacobi (rotations) method for solving the symmetric eigenvalue problem
  1. The classical Jacobi (rotations) method with pivoting for symmetric matrices
  2. Serial Jacobi (rotations) method for symmetric matrices
  3. Serial Jacobi (rotations) method with thresholding for symmetric matrices
3. Lanczos algorithm
  1. Lanczos algorithm in an algorithm (with Householder orthogonalization)

### 1.5 Algebra of polynomials

1. Horner's method

### 2 Algorithms on lists and arrays

#### 2.1 Search algorithms

1. Linear search: Finding an item in an arbitrary list:  $O(n)$
2. Binary search: Finding the position of a target value within a sorted array:  $O(\log(n))$

#### 2.2 Sorting algorithms

1. Binary tree sort
2. Bubble sort
3. Merge sort (serial and parallel variants)

#### 2.3 Graph algorithms

1. Graph traversal
  1. Breadth-First Search (BFS)
  2. Depth-First Search (DFS)
  3. Single Source Shortest Path (SSSP)
    1. Breadth-First Search (BFS) (for unweighted graphs)
    2. Dijkstra's algorithm
    3. Bellman-Ford algorithm
    4. Shortest path algorithm
  4. All Pairs Shortest Path (APSP)
    1. Johnson's algorithm
    2. Floyd-Warshall algorithm
  5. Transitive closure of a directed graph
    1. Warshall's algorithm
  6. Longest Common Subsequence (LCS)
    1. Construction of the minimum spanning tree (MST)
      1. Kruskal's algorithm
      2. Prim's algorithm
      3. Boruvka's algorithm
      4. Reverse-delete algorithm
2. Search for isomorphic subgraphs
  1. Ullmann's algorithm
  2. FT algorithm
3. Graph connectivity
  1. Tarjan-Liskov algorithm for finding the connected components
  2. DFS algorithm
  3. Tarjan's strongly connected components algorithm
  4. DFS algorithm for finding the strongly connected components
  5. Tarjan's biconnected components algorithm
  6. Tarjan-Liskov biconnected components algorithm
  7. Tarjan's algorithm for finding the bridges of a graph
  8. Flow connectivity of a graph
  9. Edmonds's edge connectivity algorithm
4. Finding maximal flow in a transportation network
  1. Ford-Fulkerson algorithm
  2. Preflow-Push algorithm
5. Finding minimal cost flow in a transportation network
  1. Cost-scaling algorithm
  2. Network simplex algorithm
6. Shortest path problems
  1. Hungarian algorithm
  2. Auction algorithm
  3. Successive shortest path algorithm
  4. Successive shortest path algorithm

Graph500

### 3 Computational geometry

1. Finding the diameter of a point set
2. Finding the convex hull of a point set
3. Finding the intersection of two line segments
4. Voronoi diagram
5. Point-in-polygon problem
6. Convex polygon intersection
7. Separated polygon intersection

### 3.1 Computer graphics

1. Line drawing algorithms: approximating a line segment discrete graphical media
2. Drawing the visible parts of a three-dimensional scene
3. Ray tracing: rendering realistic images
4. Spatial partitioning: expanding the illumination and reflection from other objects

### 4 Computer analysis and modeling

#### 4.1 Computer benchmarks

1. High Performance Conjugate Gradients (HPCG) benchmark
2. Linpack benchmark

#### 4.2 Algorithms of quantum system simulation

1. Algorithms of quantum computation simulation
  1. Single-qubit transform of a state vector
  2. Two-qubit transform of a state vector
  3. Quantum Fourier transform simulation

HPCG

# Algorithm classification and comparison of computers (what can AlgoWiki add to the standard three lists?)

## Algorithm classification

### 1 Linear algebra problems

#### 1.1 Matrix and vector operations

##### 1.1.1 Vector operations

- 1. **Dot product**
  1. **Dot product**
  2. **Parallel prefix scan algorithm using dot product**
  3. **Uniform norm of a vector (Fast Fourier Transform, serial/parallel)**
  4. **Dependency**
  5. **The serial-parallel summation method**

List8

##### 1.1.2 Matrix-vector operations

##### 1.1.2.1 Multiplying a nontrigonal matrix by a vector

- 1. **Dense matrix-vector multiplication**

##### 1.1.2.2 Multiplying a matrix of special form by a vector

- 1. **Fourier transform**
  1. **Fast Fourier transform for complex dimension**
    - 1. **Fast Fourier transform for complex values**
      - 1. **Cooley-Tukey, Fast Fourier Transform, radix-2 case**
      - 2. **Fast Fourier transform for even complex values**
    - 2. **Fast Fourier transform for complex dimension with real prime divisors (DFFT)**
  2. **Fast Fourier transform for prime dimension**

List1

##### 1.1.3 Matrix operations

- 1. **Dense matrix multiplication**
  1. **Dense matrix multiplication (serial version for real matrices)**
  2. **Strassen's algorithm**

#### 1.2 Matrix decompositions

##### 1.2.1 Matrix decomposition problem

- 1. **Triangular decomposition**
  1. **LU decomposition using Gaussian elimination (with pivoting)**
  2. **LU decomposition via Gaussian elimination**
    1. **Compositional scheme for Gaussian elimination and its modifications**
      - 1. **Compositional scheme for Gaussian elimination (Triangular matrix)**
        - 1. **Gaussian elimination, compositional scheme for triangular matrices, serial version**
        - 2. **Serial doubling algorithm for the LU decomposition of a triangular matrix**
        - 3. **Serial-parallel algorithm for the LU decomposition of a triangular matrix**
    2. **LU decomposition using Gaussian elimination with pivoting**
      - 1. **Gaussian elimination with column pivoting**
      - 2. **Gaussian elimination with row pivoting**
      - 3. **Gaussian elimination with diagonal pivoting**
      - 4. **Gaussian elimination with complete pivoting**

List14

- 2. **Cholesky method**
  - 1. **Cholesky decomposition**

Top500

##### 1.2.2 Available triangular decompositions for matrices of special form

- 1. **Unitary-triangular factorizations**
  1. **QR decomposition of dense nontrigonal matrices**
    - 1. **Givens (rotations) method for the QR decomposition of a matrix**
      - 1. **Givens method**
    - 2. **Householder (reflections) method for the QR decomposition of a matrix**
    - 3. **Householder (reflections) method for the QR decomposition of a square matrix, real/complex version**
  2. **Orthogonalization method**
    - 1. **Classical orthogonalization method**
    - 2. **Orthogonalization method with reorthogonalization**
  3. **Triangular decomposition of a Gram matrix**

List12

##### 1.2.3 Reducing matrices to compact forms

- 1. **Unitary reduction to Hessenberg form**
  1. **Householder (reflections) method for reducing a matrix to Hessenberg form**
  2. **Givens (rotations) method for reducing a matrix to Hessenberg form**
    - 1. **Classical Givens (rotations) method for reducing a matrix to Hessenberg form**
- 2. **Unitary reduction to tridiagonal form**
  1. **Householder (reflections) method for reducing a matrix to tridiagonal form**
  2. **Givens (rotations) method for reducing a complex Hermitian matrix to symmetric tridiagonal form**

List5

##### 1.2.4 Reducing matrices to compact forms

- 1. **Unitary reduction to Schur form**
  1. **Householder (reflections) method for reducing a matrix to Schur form**
  2. **Givens (rotations) method for reducing a complex Hermitian matrix to symmetric tridiagonal form**
- 2. **Unitary reduction to Schur form**
  1. **Householder (reflections) method for reducing a matrix to Schur form**
  2. **Givens (rotations) method for reducing a complex Hermitian matrix to symmetric tridiagonal form**

### 1.3 Solving systems of linear algebraic equations

#### 1.3.1 Direct methods

- 1. **Unpack benchmark**
- 2. **Classes of a special form**
  1. **Triangular matrices**
    - 1. **Forward substitution**
    - 2. **Backward substitution**
    - 3. **Serial algorithms**

List3

- 3. **Forward and backward substitution for banded matrices**
  1. **Serial doubling algorithm for solving banded LU, LU, LU**
  2. **Serial-parallel variation of the backward substitution**
  3. **Serial-parallel variation of the backward substitution**

#### 1.3.2 Methods for solving banded LU, LU, LU

- 1. **Methods based on the conventional LU decomposition**
  1. **Thomas algorithm**
    - 1. **Thomas algorithm, polynomial version**
    - 2. **Revised Thomas algorithm, polynomial version**
  2. **Serial doubling algorithm**
    - 1. **Serial doubling algorithm for the LU decomposition of banded matrices**
    - 2. **Serial doubling algorithm for solving banded LU, LU, LU**
    - 3. **Serial-parallel variation of the backward substitution**

List13

- 2. **Other methods**
  1. **Reduction method**
    - 1. **Complex reduction method**
    - 2. **Reduction method repeated for a new right-hand side**
  2. **Two-sided Thomas algorithm**
    - 1. **Two-sided Thomas algorithm, polynomial version**
    - 2. **Revised two-sided Thomas algorithm, polynomial version**
  3. **Cyclic reduction**
    - 1. **Complex cyclic reduction**
    - 2. **Cyclic reduction repeated for a new right-hand side**

- 3. **Methods for solving block triangular matrices**
  1. **Block forward substitution (real version)**
  2. **Block backward substitution (real version)**

List7

- 4. **Methods for solving block banded matrices**
  1. **Forward and backward substitution for block banded matrices**
  2. **Serial doubling algorithm for solving block banded LU, LU, LU**
  3. **Serial-parallel variation of the block backward substitution for solving block banded matrices**

- 5. **Methods for solving block LU, LU, LU**
  1. **Methods based on the conventional LU decomposition**
    - 1. **Block Thomas algorithm**
    - 2. **Serial doubling algorithm**
  2. **Serial-parallel method for solving block systems of linear algebraic equations based on the LU decomposition and backward substitution**

- 2. **Other methods**
  1. **Two-sided Thomas algorithm, block version**
  2. **Block cyclic reduction**
  3. **Block bordering method**

- 3. **Solving systems of linear algebraic equations with coefficient matrices of special form**
  1. **Serial doubling algorithm for solving linear algebraic equations**
  2. **Serial-parallel variation of the block backward substitution for solving block banded matrices**

List4

- 1. **Methods for solving block LU, LU, LU**
  1. **Methods based on the conventional LU decomposition**
    - 1. **Block Thomas algorithm**
    - 2. **Serial doubling algorithm**
  2. **Serial-parallel method for solving block systems of linear algebraic equations based on the LU decomposition and backward substitution**

#### 1.4 Solving eigenvalue problems

##### 1.4.1 Eigenvalue decomposition (finding eigenvalues and eigenvectors)

- 1. **QR algorithm**
  1. **QR algorithm as implemented in SC/LAPACK**
    - 1. **Classical (orthogonal) Householder (reflections) method for reducing a matrix to Hessenberg form**
    - 2. **Hessenberg QR algorithm as implemented in SC/LAPACK**
  2. **Symmetric QR algorithm as implemented in SC/LAPACK**
    - 1. **Householder (reflections) method for reducing a symmetric matrix to tridiagonal form**
    - 2. **Symmetric tridiagonal QR algorithm as implemented in SC/LAPACK**
  3. **QR algorithm for complex Hermitian matrices as implemented in SC/LAPACK**
    - 1. **Householder (reflections) method for reducing complex Hermitian matrix to symmetric tridiagonal form**
    - 2. **Symmetric tridiagonal QR algorithm as implemented in SC/LAPACK**

List9

- 2. **The Jacobi (rotations) method for solving the symmetric eigenvalue problem**
  1. **The classical Jacobi (rotations) method with pivoting for symmetric matrices**
  2. **Serial Jacobi (rotations) method for symmetric matrices**
  3. **Serial Jacobi (rotations) method with thresholding for symmetric matrices**

- 3. **Lanczos algorithm**
  1. **Lanczos algorithm in a vector form (with reorthogonalization)**

List15

- 2. **Partial eigenvalue problem**
  1. **Method of deflation**
  2. **Singular value decomposition (finding singular values and singular vectors)**
    - 1. **Jacobi (rotations) method for finding singular values**
      - 1. **Serial Jacobi (rotations) method for finding singular values**
      - 2. **Jacobi method with a specific choice of rotations for finding singular values**
    - 2. **QR algorithm as applied to singular value decomposition arrays**

List11

##### 1.5 Algebra of polynomials

- 1. **Horner method**

### 2 Algorithms on lists and arrays

#### 2.1 Search algorithms

- 1. **Linear search** (Finding an item in an arbitrary list:  $O(n)$ )
- 2. **Binary search** (Finding the position of a target value within a sorted array:  $O(\log(n))$ )

#### 2.2 Sorting algorithms

- 1. **Binary insertion**
- 2. **Bubble sort**
- 3. **Insertion sort (serial and parallel variants)**

#### 2.3 Graph algorithms

- 1. **Graph traversal**
  1. **Breadth-First Search (BFS)**
  2. **Depth-First Search (DFS)**
  3. **Single Source Shortest Path (SSSP)**
    - 1. **Breadth-First Search (BFS) (for unweighted graphs)**
    - 2. **Dijkstra's algorithm**
    - 3. **Bellman-Ford algorithm**
    - 4. **Shortest path algorithm**
  4. **All Pairs Shortest Path (APSP)**
    - 1. **Floyd-Warshall algorithm**
    - 2. **Transitive closure of a directed graph**
      - 1. **Warshall's algorithm**
    - 3. **Longest Shortest Path**
    - 4. **Construction of the minimum spanning tree (MST)**
      - 1. **Kruskal's algorithm**
      - 2. **Prim's algorithm**
      - 3. **Greedy algorithm**

Graph500

#### 2.4 Search for shortest paths

- 1. **Shortest path algorithms**
  1. **Shortest path algorithms**
    - 1. **Shortest path algorithms**
    - 2. **Shortest path algorithms**
    - 3. **Shortest path algorithms**
    - 4. **Shortest path algorithms**

List10

#### 2.5 Finding maximal flow in a transportation network

- 1. **Flow algorithms**
  1. **Flow algorithms**
  2. **Flow algorithms**
  3. **Flow algorithms**
  4. **Flow algorithms**

List2

#### 2.6 Finding maximal flow in a transportation network

- 1. **Flow algorithms**
  1. **Flow algorithms**
  2. **Flow algorithms**
  3. **Flow algorithms**
  4. **Flow algorithms**

List16

#### 2.7 Finding maximal flow in a transportation network

- 1. **Flow algorithms**
  1. **Flow algorithms**
  2. **Flow algorithms**
  3. **Flow algorithms**
  4. **Flow algorithms**

List6

#### 2.8 Finding maximal flow in a transportation network

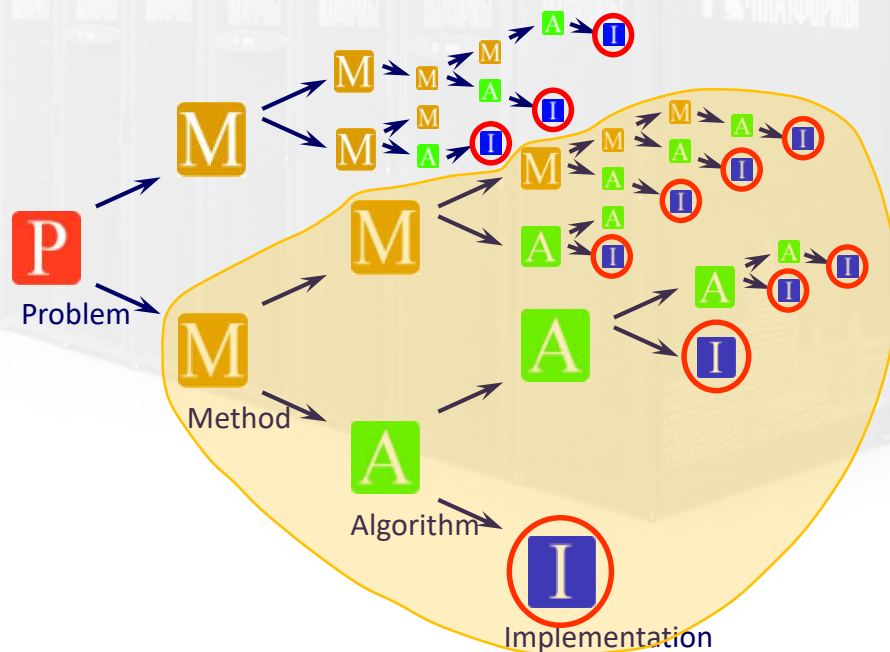
- 1. **Flow algorithms**
  1. **Flow algorithms**
  2. **Flow algorithms**
  3. **Flow algorithms**
  4. **Flow algorithms**

HPCG

# “Top500” list for the given Problem and Method



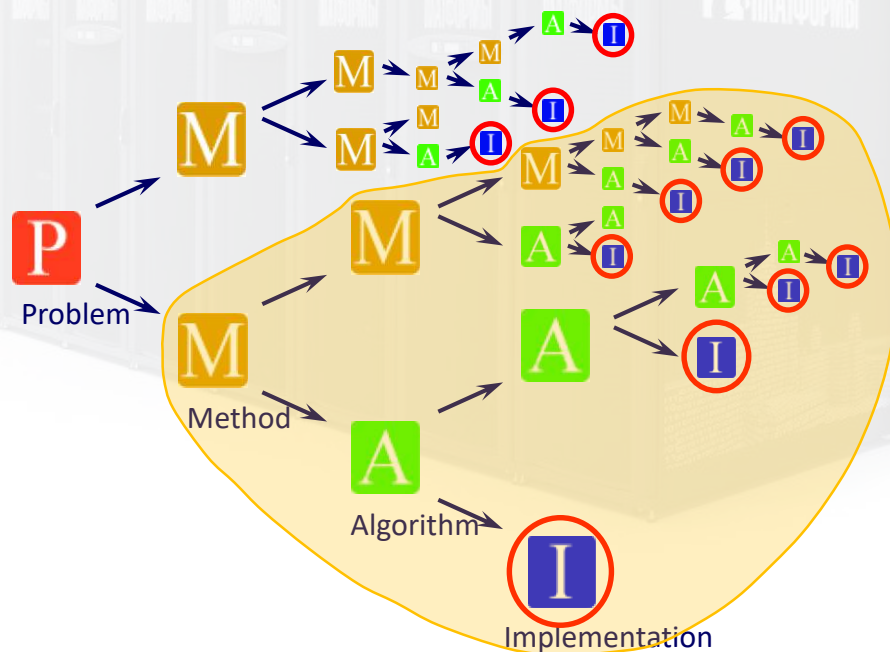
- compare different algorithms, implementations and computing platforms for the problem and method;



# “Top500” list on “*Strongly Connected Components*” (Method = *Forward-Backward*)



- compare different algorithms, implementations and computing platforms for the problem and method;



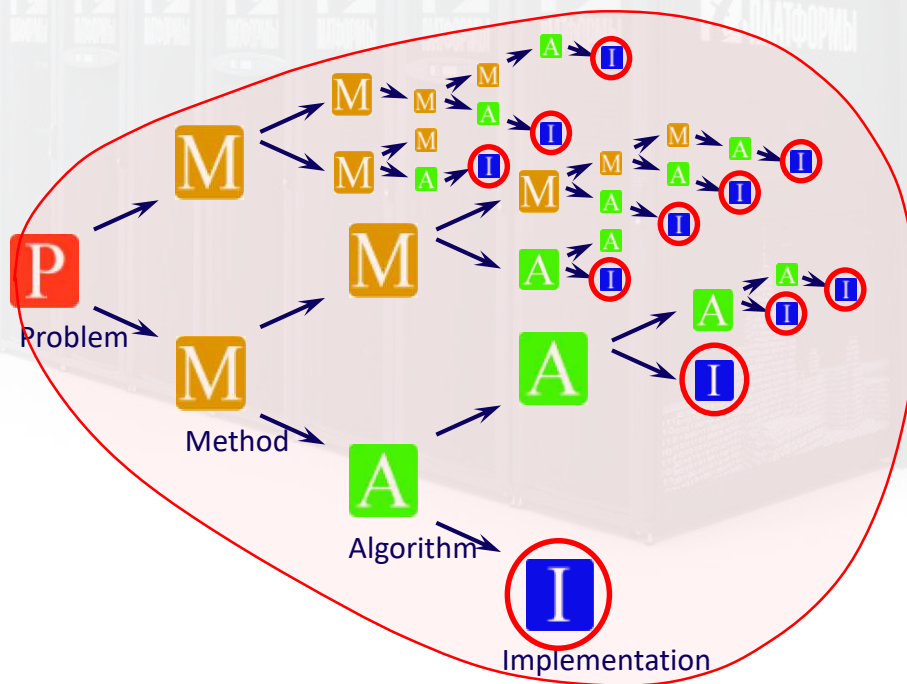
# “Top500” list on “Strongly Connected Components” (Method = Forward-Backward)

| Rating | Method           | Implementation | Platform           | MTEPS  | GraphType | GraphSize |
|--------|------------------|----------------|--------------------|--------|-----------|-----------|
| 1      | Forward-Backward | RCC for GPU    | Lomonosov-2 (P100) | 634,00 | RMAT      | 2^20      |
| 2      | Forward-Backward | RCC for GPU    | Lomonosov-2 (P100) | 620,00 | RMAT      | 2^21      |
| 3      | Forward-Backward | RCC for CPU    | Lomonosov-2        | 564,00 | RMAT      | 2^24      |
| 4      | Forward-Backward | RCC for GPU    | Lomonosov-2 (P100) | 544,00 | RMAT      | 2^22      |
| 5      | Forward-Backward | RCC for GPU    | Lomonosov-2 (P100) | 528,00 | RMAT      | 2^23      |
| 6      | Forward-Backward | RCC for CPU    | Lomonosov-2        | 498,00 | RMAT      | 2^26      |
| 7      | Forward-Backward | RCC for CPU    | Lomonosov-2        | 497,00 | RMAT      | 2^25      |
| 8      | Forward-Backward | RCC for CPU    | Lomonosov-2        | 486,00 | RMAT      | 2^27      |
| 9      | Forward-Backward | RCC for GPU    | Lomonosov-2 (P100) | 456,00 | RMAT      | 2^25      |
| 10     | Forward-Backward | RCC for GPU    | Lomonosov-2 (P100) | 453,00 | RMAT      | 2^24      |
| 11     | Forward-Backward | RCC for CPU    | Lomonosov-2        | 452,00 | RMAT      | 2^22      |
| 12     | Forward-Backward | RCC for CPU    | Lomonosov-2        | 440,24 | SSCA-2    | 2^21      |
| 13     | Forward-Backward | RCC for CPU    | Lomonosov-2        | 432,00 | RMAT      | 2^23      |
| 14     | Forward-Backward | RCC for CPU    | Lomonosov-2        | 426,00 | RMAT      | 2^21      |
| 15     | Forward-Backward | RCC for GPU    | Lomonosov-2 (P100) | 426,00 | RMAT      | 2^26      |
| 16     | Forward-Backward | RCC for CPU    | Lomonosov-2        | 418,00 | RMAT      | 2^20      |
| 17     | Forward-Backward | PBGL MPI       | IBM BlueGene/P     | 232,86 | RMAT      | 2^20      |
| 18     | Forward-Backward | RCC for GPU    | Lomonosov-2        | 195,00 | RMAT      | 2^18      |
| 19     | Forward-Backward | PBGL MPI       | Lomonosov          | 91,07  | RMAT      | 2^21      |
| 20     | Forward-Backward | RCC for CPU    | Lomonosov-2        | 55,44  | RMAT      | 2^18      |
| 21     | Forward-Backward | RCC for CPU    | IBM Regatta        | 53,60  | SSCA-2    | 2^18      |
| 22     | Forward-Backward | PBGL MPI       | IBM BlueGene/P     | 45,75  | RMAT      | 2^20      |
| 23     | Forward-Backward | RCC for GPU    | Lomonosov          | 44,78  | RMAT      | 2^16      |
| 24     | Forward-Backward | RCC for CPU    | Lomonosov          | 42,00  | RMAT      | 2^22      |
| 25     | Forward-Backward | RCC for CPU    | Lomonosov          | 41,00  | RMAT      | 2^20      |
| 26     | Forward-Backward | RCC for CPU    | IBM Regatta        | 36,90  | RMAT      | 2^18      |
| 27     | Forward-Backward | RCC for CPU    | Lomonosov          | 32,54  | RMAT      | 2^20      |
| 28     | Forward-Backward | PBGL MPI       | IBM BlueGene/P     | 13,39  | SSCA-2    | 2^16      |
| 29     | Forward-Backward | PBGL MPI       | IBM BlueGene/P     | 13,12  | SSCA-2    | 2^18      |
| 30     | Forward-Backward | RCC for CPU    | Lomonosov          | 10,05  | SSCA-2    | 2^20      |
| 31     | Forward-Backward | RCC for CPU    | Lomonosov          | 9,20   | SSCA-2    | 2^18      |
| 32     | Forward-Backward | RCC for CPU    | Lomonosov          | 8.30   | SSCA-2    | 2^20      |

# “Top500” list for the given Problem



- compare different ways of solving the problem;

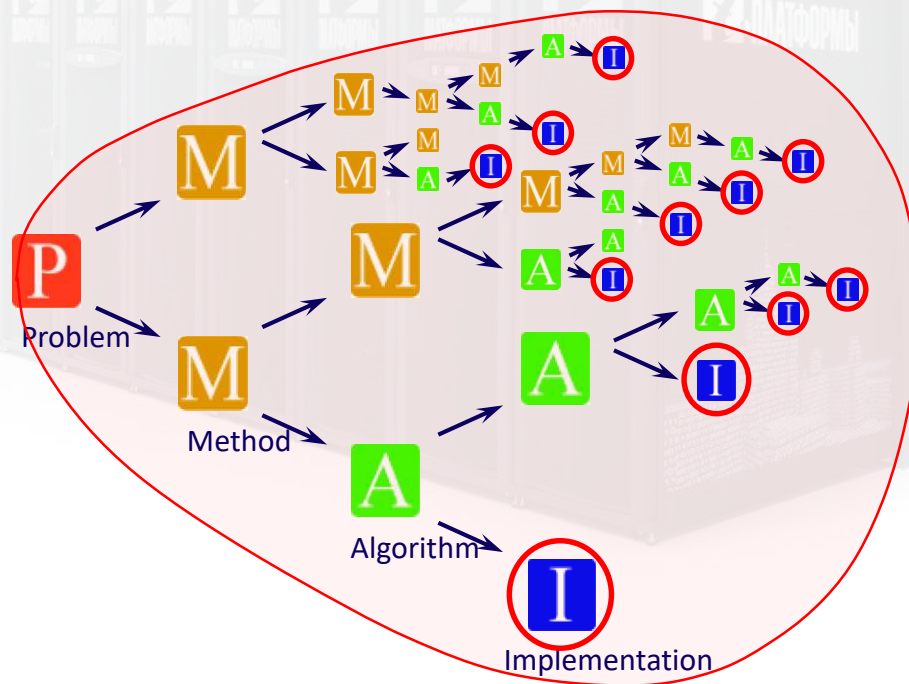


# “Top500” list on “*Strongly Connected Components*”

(various methods, algorithms, implementations, computers)



- compare different ways of solving the problem;



“Top500” list on “**Strongly Connected Components**”  
*(various methods, algorithms, implementations, computers)*

| Rating | Method           | Implementation | Platform           | MTEPS   | GraphType | GraphSize |
|--------|------------------|----------------|--------------------|---------|-----------|-----------|
| 1      | Shiloach-Vishkin | Ligra          | Lomonosov-2        | 1307,00 | RMAT      | 2^26      |
| 2      | Shiloach-Vishkin | Ligra          | Lomonosov-2        | 986,00  | RMAT      | 2^23      |
| 3      | Shiloach-Vishkin | Ligra          | Lomonosov-2        | 947,00  | RMAT      | 2^22      |
| 4      | Shiloach-Vishkin | Ligra          | Lomonosov-2        | 894,00  | RMAT      | 2^24      |
| 5      | Shiloach-Vishkin | Ligra          | Lomonosov-2        | 864,00  | RMAT      | 2^25      |
| 6      | Shiloach-Vishkin | Ligra          | Lomonosov-2        | 830,00  | RMAT      | 2^20      |
| 7      | Shiloach-Vishkin | Ligra          | Lomonosov-2        | 782,00  | RMAT      | 2^21      |
| 8      | Forward-Backward | RCC for GPU    | Lomonosov-2 (P100) | 634,00  | RMAT      | 2^20      |
| 9      | Forward-Backward | RCC for GPU    | Lomonosov-2 (P100) | 620,00  | RMAT      | 2^21      |
| 10     | Forward-Backward | RCC for CPU    | Lomonosov-2        | 564,00  | RMAT      | 2^24      |
| 11     | Shiloach-Vishkin | GAP            | Lomonosov-2        | 547,00  | RMAT      | 2^20      |
| 12     | Forward-Backward | RCC for GPU    | Lomonosov-2 (P100) | 544,00  | RMAT      | 2^22      |
| 13     | Forward-Backward | RCC for GPU    | Lomonosov-2 (P100) | 528,00  | RMAT      | 2^23      |
| 14     | Forward-Backward | RCC for CPU    | Lomonosov-2        | 498,00  | RMAT      | 2^26      |
| 15     | Forward-Backward | RCC for CPU    | Lomonosov-2        | 497,00  | RMAT      | 2^25      |
| 16     | Forward-Backward | RCC for CPU    | Lomonosov-2        | 486,00  | RMAT      | 2^27      |
| 17     | Shiloach-Vishkin | GAP            | Lomonosov-2        | 480,00  | RMAT      | 2^22      |
| 18     | Forward-Backward | RCC for GPU    | Lomonosov-2 (P100) | 456,00  | RMAT      | 2^25      |
| 19     | Forward-Backward | RCC for GPU    | Lomonosov-2 (P100) | 453,00  | RMAT      | 2^24      |
| 20     | Forward-Backward | RCC for CPU    | Lomonosov-2        | 452,00  | RMAT      | 2^22      |
| 21     | Forward-Backward | RCC for CPU    | Lomonosov-2        | 440,24  | SSCA-2    | 2^21      |
| 22     | Forward-Backward | RCC for CPU    | Lomonosov-2        | 432,00  | RMAT      | 2^23      |
| 23     | Forward-Backward | RCC for CPU    | Lomonosov-2        | 426,00  | RMAT      | 2^21      |
| 24     | Forward-Backward | RCC for GPU    | Lomonosov-2 (P100) | 426,00  | RMAT      | 2^26      |
| 25     | Forward-Backward | RCC for CPU    | Lomonosov-2        | 418,00  | RMAT      | 2^20      |
| 26     | Shiloach-Vishkin | GAP            | Lomonosov-2        | 387,00  | RMAT      | 2^23      |
| 27     | Shiloach-Vishkin | GAP            | Lomonosov-2        | 335,00  | RMAT      | 2^21      |
| 28     | Forward-Backward | PBGL MPI       | IBM BlueGene/P     | 232,86  | RMAT      | 2^20      |
| 29     | Shiloach-Vishkin | GAP            | Lomonosov-2        | 231,00  | RMAT      | 2^24      |
| 30     | Forward-Backward | RCC for GPU    | Lomonosov-2        | 195,00  | RMAT      | 2^18      |
| 31     | Shiloach-Vishkin | GAP            | Lomonosov-2        | 180,00  | RMAT      | 2^25      |
| 32     | Shiloach-Vishkin | GAP            | Lomonosov-2        | 174,00  | RMAT      | 2^26      |

AlgoWiki data:  
 submitted by  
 different people  
 from everywhere...



# AlgoWiki: an easy step back to analyze "Top500" results (comprehensive evaluation of computing platforms)

SSSP problem:

| Method         | Implementation | Computing Platform              | MTEPS | GraphType | GraphSize |
|----------------|----------------|---------------------------------|-------|-----------|-----------|
| Bellman-Ford   | RCC for CPU    | Lomonosov-2                     | 418,0 | RMAT      | 2^20      |
| Bellman-Ford   | Graph500 MPI   | Lomonosov                       | 350,0 | RMAT      | 2^20      |
| Bellman-Ford   | RCC for CPU    | Lomonosov-2                     | 204,1 | RMAT      | 2^20      |
| Dijkstra's     | PBGL MPI       | Cluster / "Angara" interconnect | 150,0 | SSCA-2    | 2^20      |
| Delta-Stepping | PBGL MPI       | Lomonosov                       | 124,1 | SSCA-2    | 2^21      |
| Bellman-Ford   | Graph500 MPI   | Lomonosov                       | 120,0 | RMAT      | 2^20      |
| Dijkstra's     | PBGL MPI       | IBM BlueGene/P                  | 8,9   | SSCA-2    | 2^20      |
| Dijkstra's     | PBGL MPI       | Lomonosov                       | 5,3   | SSCA-2    | 2^21      |
| Delta-Stepping | PBGL MPI       | IBM BlueGene/P                  | 3,8   | SSCA-2    | 2^20      |



AlgoWiki Page: Classification

## Dijkstra's algorithm

Primary authors of this descriptor: A.N. Danyil, Vad V. Yevodkin (Section 2.2)

**Contents [hide]**

- 1. Properties and structure of the algorithm:
  - 1.1 General description of the algorithm
  - 1.2 Mathematical description of the algorithm
  - 1.3 Computational kernel of the algorithm
  - 1.4 Stack structure of the algorithm
  - 1.5 Implementation scheme of the serial algorithm
  - 1.6 Serial complexity of the algorithm

### 1.6 Serial complexity of the algorithm

The serial complexity of the algorithm is  $O(C_1 m + C_2 n)$ , where:

- $C_1$  is the number of operations for decreasing the distance to a node.
- $C_2$  is the number of operations for calculating minima.

The original Dijkstra's algorithm used lists as an internal data structure. For such lists,  $C_1 = O(1)$ ,  $C_2 = O(n)$ , and the total complexity is  $O(n^2)$ .

### 1.8 Parallelization resource of the algorithm

Dijkstra's algorithm admits an efficient parallelization [2] its average execution time is  $O(n^{1/3} \ln n)$ , and the computational complexity is  $O(n \ln n + m)$ .

The algorithm of  $\Delta$ -stepping can be regarded as a parallel version of Dijkstra's algorithm.

# *General methodology to compare computing platforms* (using any algorithm)

problems for evaluation of computer platforms



AlgoWiki as an extension of the Top500 Methodology ?

Well-known  
theoretical potential

Well-described  
community experience



V.P.IVANNIKOV MEMORIAL WORKSHOP

*Thank you !*

*AlgoWiki as an extension of the Top500 Methodology*

*Algowiki-Project.org*

*May, 3<sup>rd</sup> 2018, Erevan, Armenia*