The 8th International Conference
"Distributed Computing and GRID-Technologies in Science and Education"

Building Up Intelligible Parallel Computing World

Prof. Vladimir Voevodin
Deputy Director, Research Computing Center, MSU
Head of Department on Supercomputers and Quantum Informatics, CMC, MSU

voevodin@parallel.ru

September, 10th 2018, LIT JINR, Dubna

Supercomputer Technologies Are Everywhere…

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

MSU Petascale Facilities:
Supercomputers Lomonosov-2 and Lomonosov

MSU Supercomputing Center today:
Users: 2955
Projects: 880

MSU Faculties / Institutes : 21
Institutes of RAS : 95
Russian Universities: 102

Supercomputer Technologies
Are Everywhere…

MSU Supercomputer Platforms

Technological background:
Intel Xeon 4/6/10/12… cores
SMP node on 128 cores
Intel Xeon Phi (KNL)
NVIDIA 2070 / 2090 / K40 / P100
IBM Power 8 / IBM Blue Gene/P
Memory per node: from 12GB up to 2TB

Diversity is great…

Existing methodologies to compare computing platforms
(Top500, Graph500, HPCG)

High Performance Linpack
Benchmark
Top500.org

BFS & SSSP Graph
Benchmarks
Graph500.org

High Performance Conjugate Gradients
Benchmark
hpcg-benchmark.org

Well-known theoretical
potential of algorithms

Well-described and available
community experience

Existing methodologies to compare computing platforms
(Top500, Graph500, HPCG)

TOP 500 The List.

High Performance Linpack Benchmark
Top500.org

GRAPH 500

BFS & SSSP Graph Benchmarks
Graph500.org

HPCG

High Performance Conjugate Gradients Benchmark
hpcg-benchmark.org

↓ ↓ ↓

Well-known theoretical potential of algorithms

↓ ↓ ↓

Well-described and available community experience

General methodology to compare computing platforms
(using any algorithm)

?

?

Well-known theoretical potential of algorithms

Well-described and available community experience

Well-known theoretical potential of algorithms

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

What is a description of an algorithm?

Description of algorithms

(What properties 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 mat[...]ion $A = U^*U$, 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 represntation. The essence of this decomposition consists in the implementation of formulas obtained uniquely for the elements of the matrix L from [...] features.

General Description

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 for[...]

Mathematical Description

$$l_{11} = \sqrt{a_{11}},$$

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

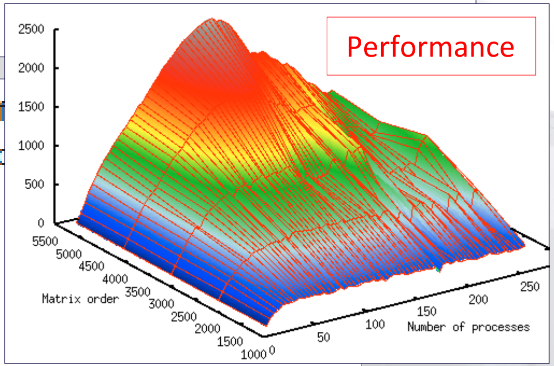
$$l_{ii} = \sqrt{a_{ii} - \sum_{p=1}^{i-1} l_{ip}^2}, \quad i \in [2, n],$$

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

The following number of operations should be perf[...] matrix of order n using a serial version of the Cho[...]

Serial Complexity

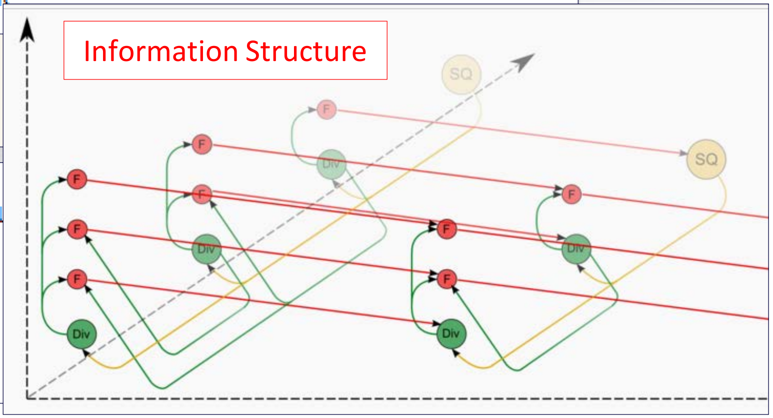
- n square roots,
- $\frac{n(n-1)}{2}$ divisiona,
- $\frac{n^3 - n}{6}$ multiplications and $\frac{n^3 - n}{6}$ additions (subtractions): the main amount of computational work



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

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

Computational Kernel



AlgoWiki

Файл Правка Вид Журнал Закладки Инструменты Справка

Open Encyclopedia of Par...

algowiki-project.org/en/Open_Encyclopedia_of_Parallel_Algorithmic_Features

Create account Log in

AlgoWiki

Page | Discussion | Read | View source | View history | Search

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

Open Encyclopedia of Parallel Algorithmic Features

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

Properties of the algorithm:

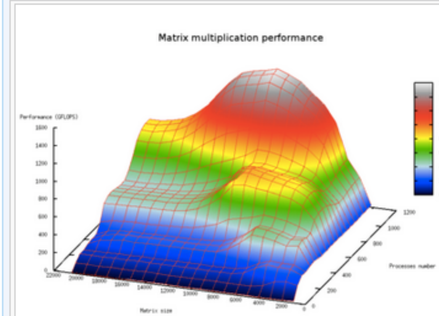
- Sequential complexity: $O(n^3)$
- Height of the parallel form: $O(n)$
- Width of the parallel form: $O(n^2)$
- Amount of input data: $\frac{n(n+1)}{2}$
- Amount of output data: $\frac{n(n+1)}{2}$

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

Today's featured picture



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

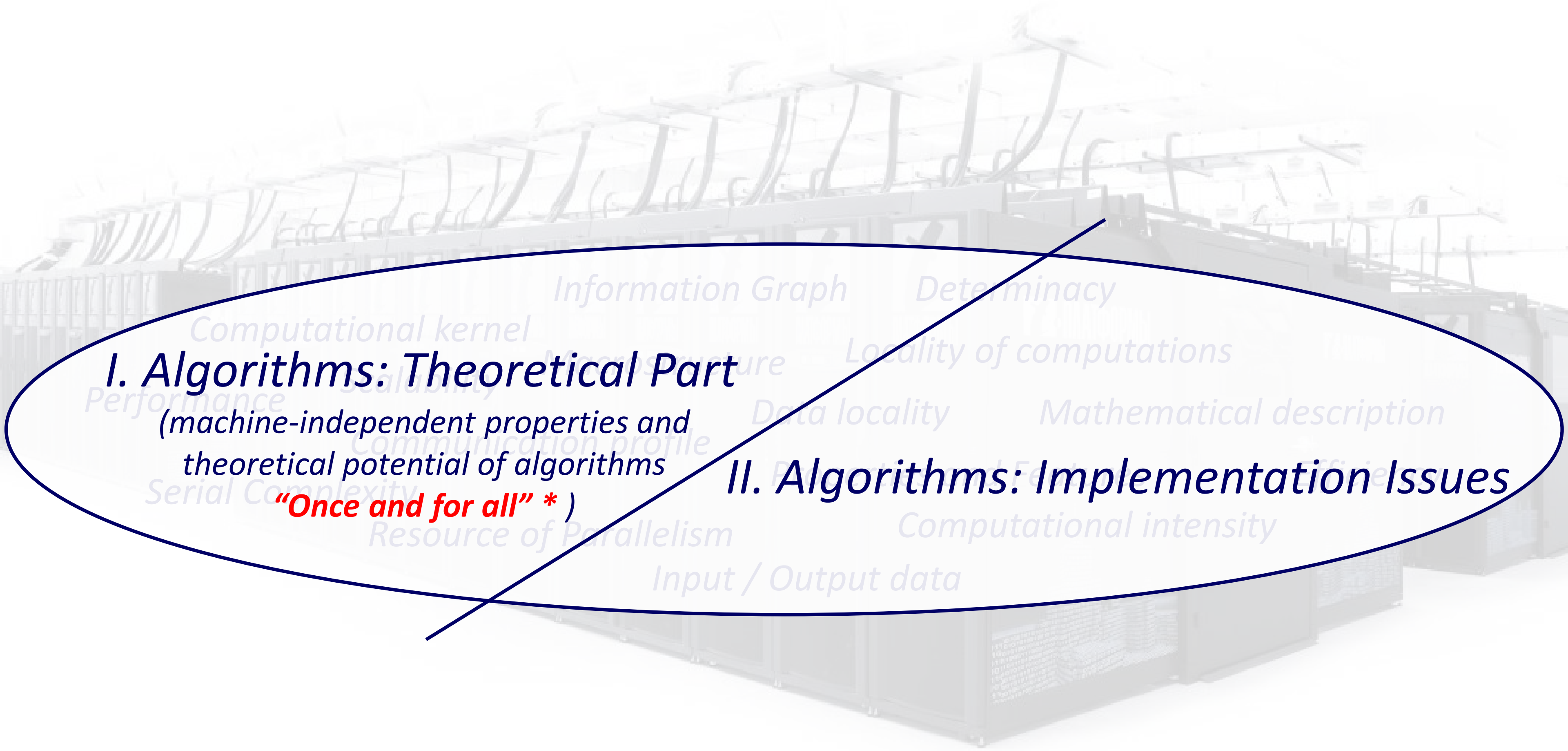
Readiness of articles

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

http://AlgoWiki-Project.org/en

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



** Changes in architectures require changes in implementations, not in algorithms!*

Unified description of algorithms in AlgoWiki

(predefined unified structure)

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 Macro structure of the algorithm
- 1.5 Implementation scheme of the serial algorithm
- 1.6 Serial complexity of the algorithm
- 1.7 Information graph
- 1.8 Parallelization resource of the algorithm
- 1.9 Input and output data of the algorithm
- 1.10 Properties of the algorithm

2 Software implementation of the algorithm
- 2.1 Implementation peculiarities of the serial algorithm
- 2.2 Locality of data and computations
- 2.3 Possible methods and considerations for parallel implementation of the algorithm
- 2.4 Scalability of the algorithm and its implementations
- 2.5 Dynamic characteristics and efficiency of the algorithm implementation
- 2.6 Conclusions for different classes of computer architecture
- 2.7 Existing implementations of the algorithm

3 References

AlgoWiki: algorithm classification
(tree structure)

1 Linear algebra problems

1.1 Matrix and vector operations

1.1.1 Vector operations

1. Pairwise summation
   1. Pairwise summation of numbers
   2. Parallel prefix scan algorithm using pairwise summation
2. Uniform norm of a vector: Real version, serial-parallel variant
3. Dot product
4. The serial-parallel summation method

1.1.2 Matrix-vector operations

1.1.2.1 Multiplying a nonsingular 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 composite dimension
      1. Fast Fourier transform for powers-of-two
         1. Cooley–Tukey Fast Fourier Transform, radix-2 case
         2. Fast Fourier transform for even powers-of-two
      2. Fast Fourier transform for composite dimension with small 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

Matrix decomposition problem

1. Triangular decompositions
   1. Gaussian elimination (finding the LU decomposition)
      1. LU decomposition using Gaussian elimination without pivoting
         1. LU decomposition via Gaussian elimination
         2. Gaussian elimination, compact scheme for tridiagonal matrices and its modifications
            1. Compact scheme for Gaussian elimination: Dense matrix
            2. Compact scheme for Gaussian elimination and its modifications: Tridiagonal matrix
               1. Gaussian elimination, compact scheme for tridiagonal matrices, serial version
               2. Stone doubling algorithm for the LU decomposition of a tridiagonal matrix
               3. Serial-parallel algorithm for the LU decomposition of a tridiagonal 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
   2. Cholesky method
      1. Cholesky decomposition
2. Available triangular decompositions for matrices of special form
3. Unitary-triangular 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
         1. Householder (reflections) method for the QR decomposition of a square matrix, real point-wise version
      3. Orthogonalization method
         1. Classical orthogonalization method
         2. Orthogonalization method with reorthogonalization
      4. Triangular decomposition of a 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
4. Reducing matrices to compact forms
   1. Unitary reductions to Hessenberg form
      1. Householder (reflections) method for reducing of a matrix to Hessenberg form
         1. Classical point-wise Householder (reflections) method for reducing a matrix to Hessenberg form
      2. Givens (rotations) method for reducing a matrix to Hessenberg form
         1. Classical point-wise Givens (rotations) method for reducing a matrix to Hessenberg form
   2. Unitary reductions to tridiagonal form
      1. Householder (reflections) method for reducing to tridiagonal form
         1. Householder (reflections) method for reducing a symmetric matrix to tridiagonal form
         2. Householder (reflections) method for reducing a complex Hermitian matrix to symmetric tridiagonal form
      2. Givens (rotations) reduction to tridiagonal form
   3. Eigenvalue decomposition (finding eigenvalues and eigenvectors)
5. Unitary non-similarity reductions to compact forms
   1. Unitary reduction to bidiagonal form
      1. Householder (reflections) reduction of a matrix to bidiagonal form
      2. Givens (rotations) reduction of a matrix to bidiagonal form
   2. Singular value decomposition
      1. Singular value decomposition (finding singular values and singular vectors)

1.3 Solving systems of linear algebraic equations

1. Direct methods
   1. Linpack benchmark
   2. Matrices of a special form
      1. Triangular matrices
         1. Forward substitution
         2. Backward substitution
         3. Bidiagonal matrices
            1. Forward and backward substitution for bidiagonal matrices
            2. Stone doubling algorithm for solving bidiagonal SLAEs
            3. Serial-parallel variant of the backward substitution
      2. Methods for solving tridiagonal SLAEs
         1. Methods based on the conventional LU decomposition
            1. Thomas algorithm
               1. Thomas algorithm, pointwise version
               2. Repeated Thomas algorithm, pointwise version
            2. Stone doubling algorithm
               1. Stone doubling algorithm for the LU decomposition of tridiagonal matrices
               2. Stone doubling algorithm for solving bidiagonal SLAEs
            3. Serial-parallel method for solving tridiagonal matrices based on the LU decomposition and backward substitutions
         2. Other methods
            1. Reduction method
               1. Complete reduction method
               2. Reduction method repeated for a new right-hand side
            2. Two-sided Thomas algorithm
               1. Two-sided Thomas algorithm, pointwise version
               2. Repeated two-sided Thomas algorithm, pointwise version
            3. Cyclic reduction
               1. Complete cyclic reduction
               2. Cyclic reduction repeated for a new right-hand side
            4. Bordering method
      3. 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. Stone doubling algorithm for solving block bidiagonal matrices
            3. Serial-parallel variant of the block backward substitution for solving block bidiagonal matrices
      4. Methods for solving block tridiagonal 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 variant
            2. Block cyclic reduction
            3. Block bordering method
   3. Solving systems of linear algebraic equations with coefficient matrices of special form whose inverses are known
2. Iterative methods for solving systems of linear algebraic equations
   1. High Performance Conjugate Gradient (HPCG) benchmark
   2. Biconjugate gradient stabilized method (BiCGStab)
   3. Kaczmarz's algorithm

1.4 Solving eigenvalue problems

1. Eigenvalue decomposition (finding eigenvalues and eigenvectors)
   1. QR algorithm
      1. QR algorithm as implemented in SCALAPACK
         1. Classical point-wise Householder (reflections) method for reducing a matrix to Hessenberg form
         2. Hessenberg QR algorithm as implemented in SCALAPACK
      2. Symmetric QR algorithm as implemented in SCALAPACK
         1. Householder (reflections) method for reducing a symmetric matrix to tridiagonal form
         2. Symmetric tridiagonal QR algorithm as implemented in SCALAPACK
      3. QR algorithm for complex Hermitian matrices as implemented in SCALAPACK
         1. Householder (reflections) method for reducing a complex Hermitian matrix to symmetric tridiagonal form
         2. Symmetric tridiagonal QR algorithm as implemented in SCALAPACK
   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 thresholds for symmetric matrices
   3. Lanczos algorithm
      1. Lanczos algorithm in exact algorithm (without reorthogonalization)
2. Partial eigenvalue problem
   1. Method of bisection
3. 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 special choice of rotations for finding singular values
   2. QR algorithm as applied to singular value decomposition a matrix

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)
2. Single Source Shortest Path (SSSP)
   1. Breadth-first search (BFS) (for unweighted graphs)
   2. Dijkstra's algorithm
   3. Bellman-Ford algorithm
   4. Δ-stepping algorithm
3. All Pairs Shortest Path (APSP)
   1. Johnson's algorithm
   2. Floyd-Warshall algorithm
4. Transitive closure of a directed graph
   1. Purdom's algorithm
5. Longest shortest path
6. Construction of the minimum spanning tree (MST)
   1. Boruvka's algorithm
   2. Kruskal's algorithm
   3. Prim's algorithm
   4. GHS algorithm
7. Search for isomorphic subgraphs
   1. Ullman's algorithm
   2. VF2 algorithm
8. Graph connectivity
   1. Shiloach-Vishkin algorithm for finding the connected components
   2. Disjoint set union
   3. Tarjan's strongly connected components algorithm
   4. DCSC algorithm for finding the strongly connected components
   5. Tarjan's biconnected components algorithm
   6. Tarjan-Vishkin biconnected components algorithm
   7. Tarjan's algorithm for finding the bridges of a graph
   8. Vertex connectivity of a graph
   9. Gabow's edge connectivity algorithm
9. Finding maximal flow in a transportation network
   1. Ford–Fulkerson algorithm
   2. Preflow-Push algorithm
10. Finding minimal-cost flow in a transportation network
11. Assignment problem
    1. Hungarian algorithm
    2. Auction algorithm
    3. Hopcroft–Karp algorithm
12. Betweenness centrality algorithm

3 Computational geometry

1. Finding the diameter of a point set
2. Finding the convex hull of a point set
3. Delaunay triangulation
4. Voronoi diagram
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 on discrete graphical media
2. Determining visible parts of a three-dimensional scene
3. Ray tracing: Rendering realistic images
4. Global illumination: Regarding direct illumination and reflection from other objects

4 Computer analysis and modeling

4.1 Computer benchmarks

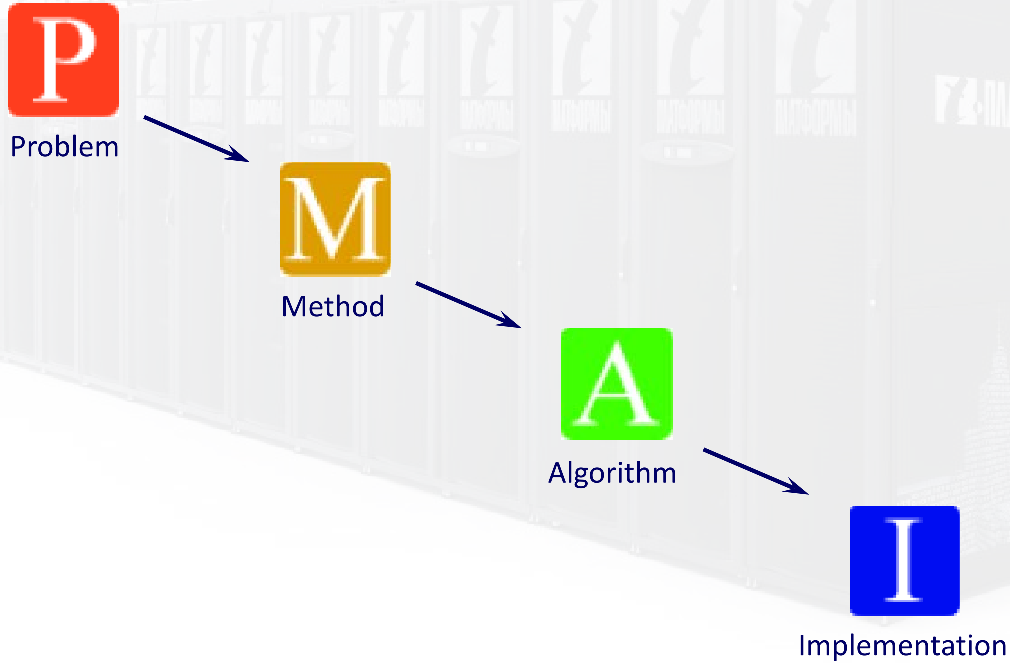
1. High Performance Conjugate Gradient (HPCG) benchmark
2. Linpack benchmark

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

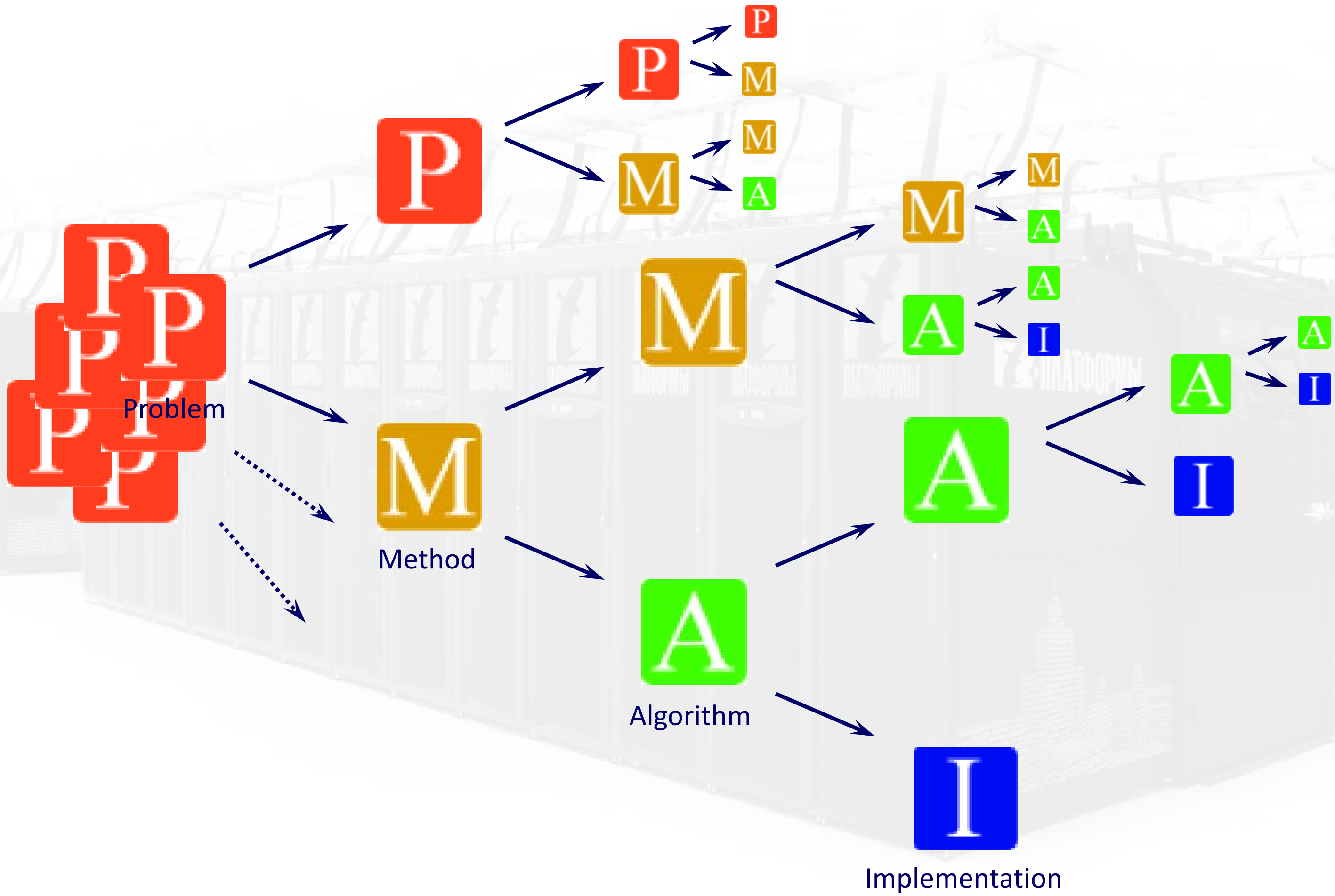
AlgoWiki: Problem – Method – Algorithm – Implementation

(What do we have for each algorithm in AlgoWiki?
An exhaustive description of the chain)



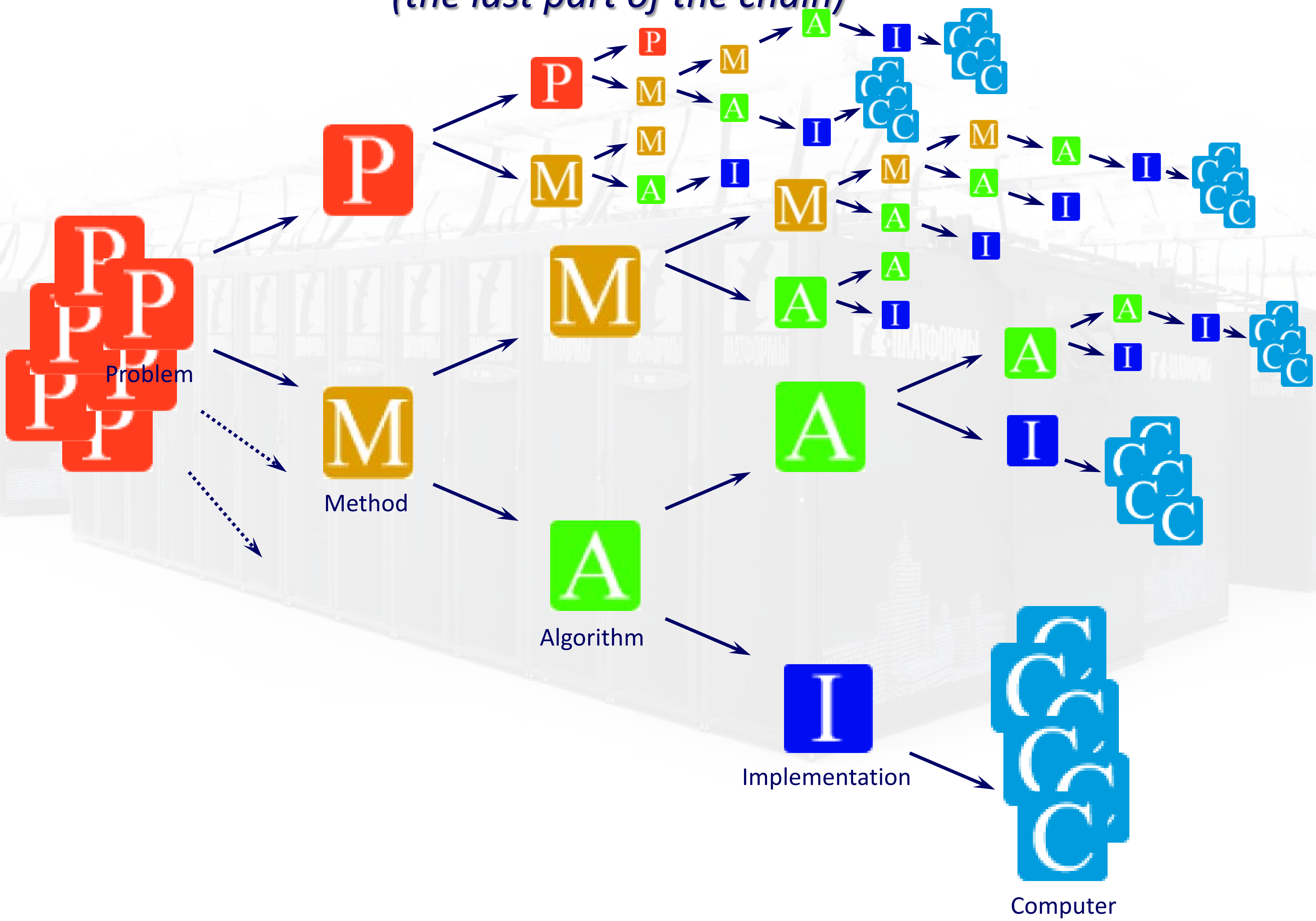
AlgoWiki: Problem – Method – Algorithm – Implementation

(the situation is more complex, diverse and sophisticated in practice)



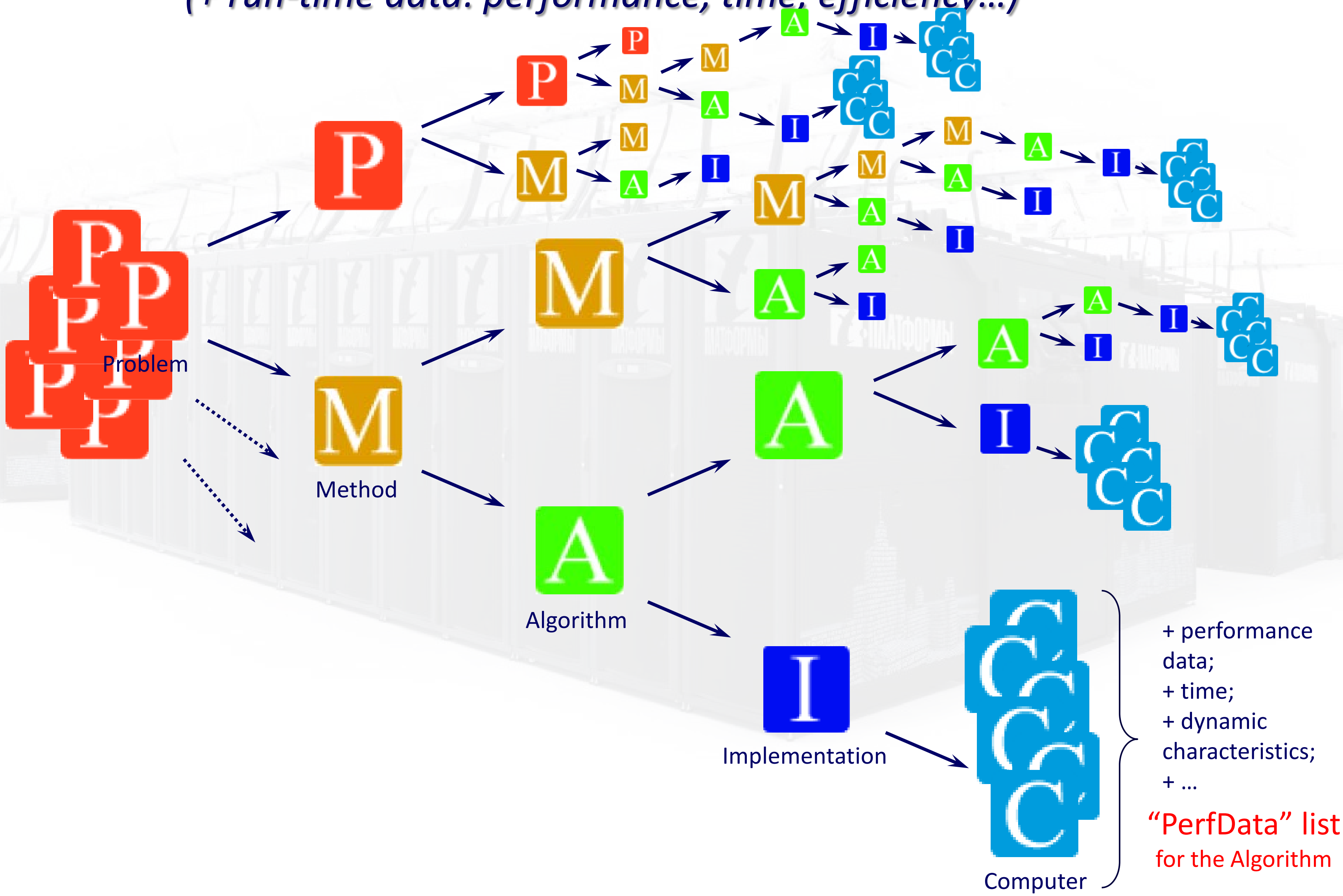
AlgoWiki: from Problems to various Implementations

(the last part of the chain)



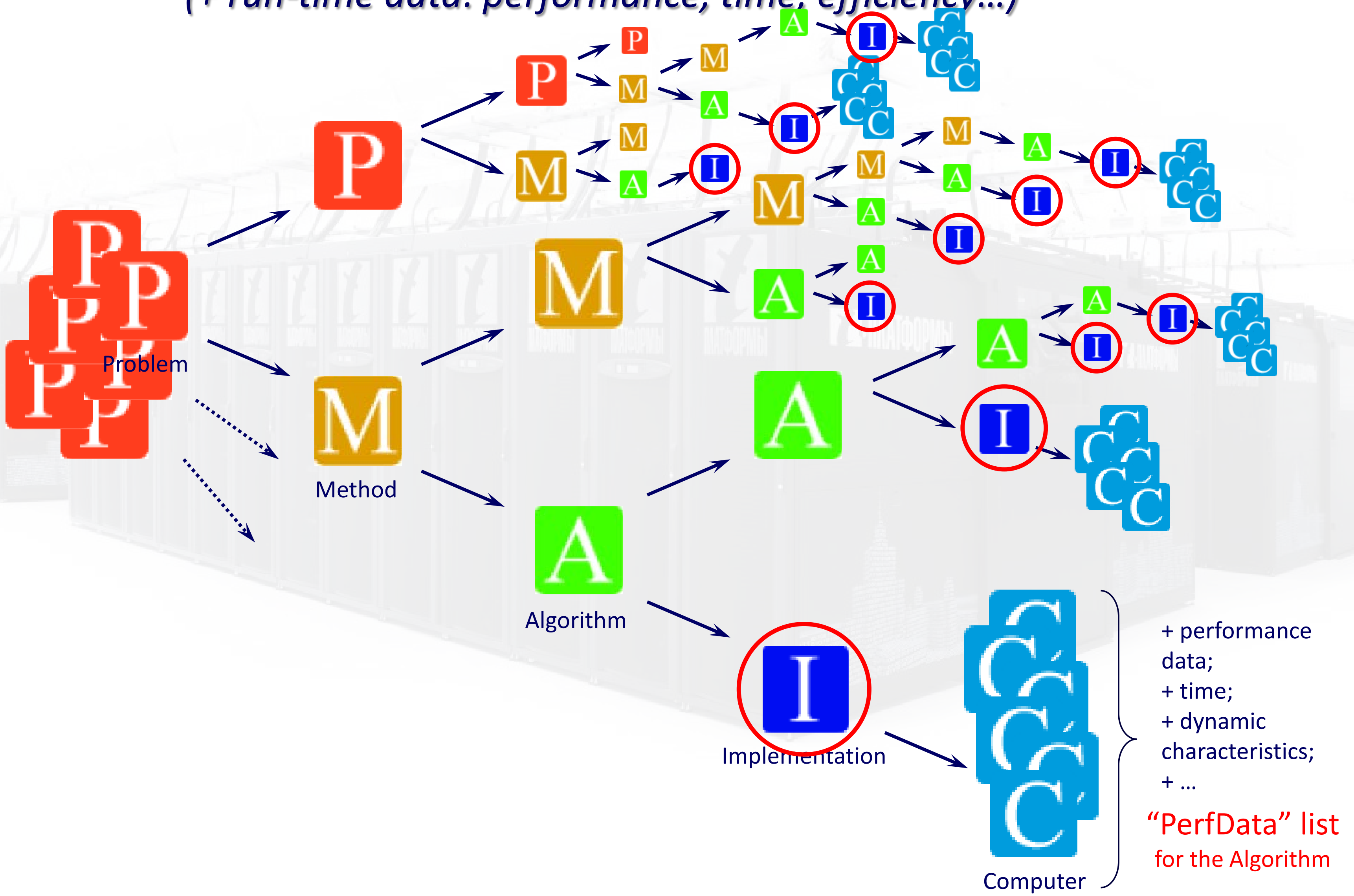
AlgoWiki: from Problems to various Implementations

(+ run-time data: performance, time, efficiency…)



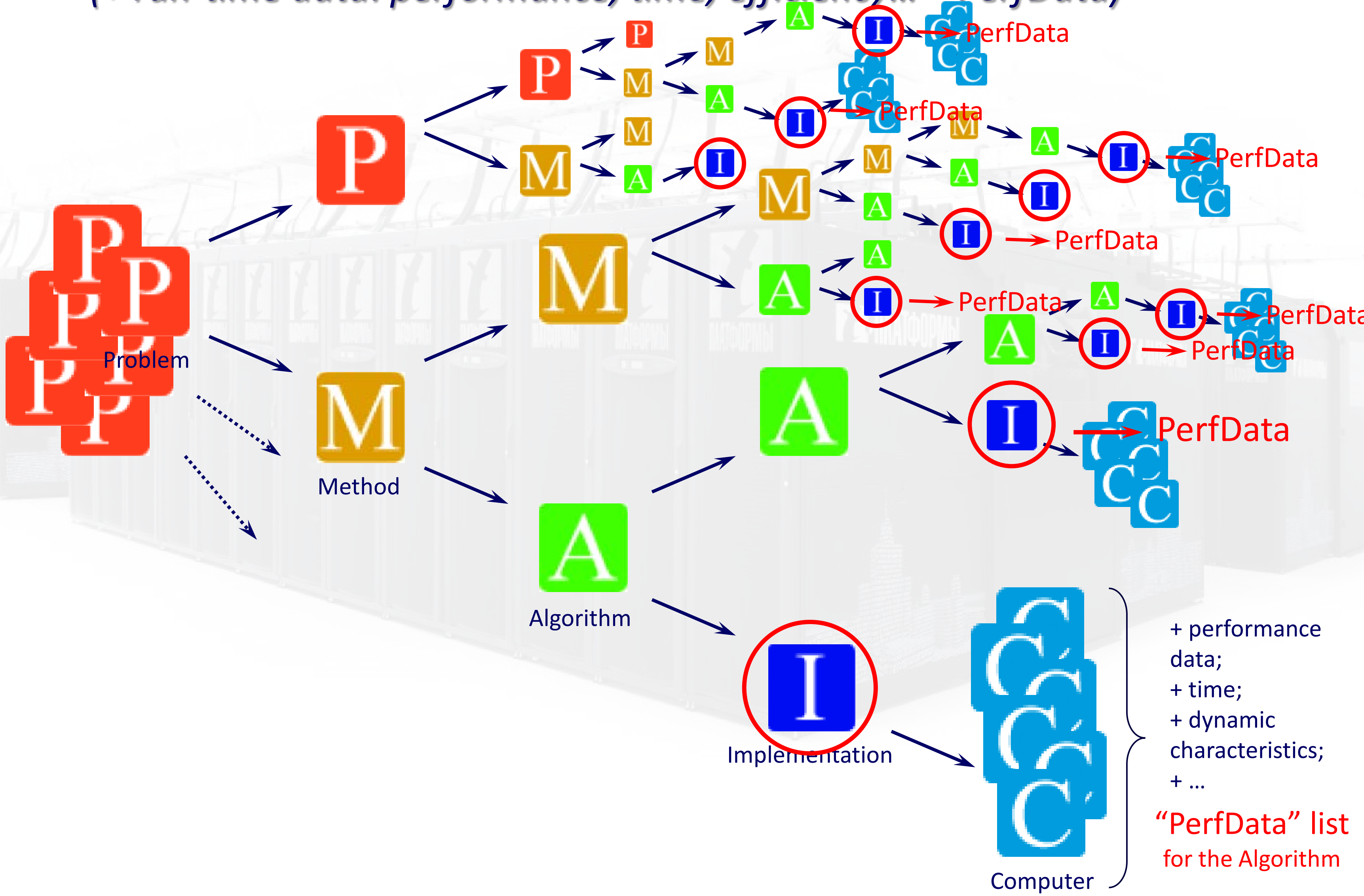
AlgoWiki: from Problems to various Implementations

(+ run-time data: performance, time, efficiency…)



AlgoWiki: from Problems to various Implementations

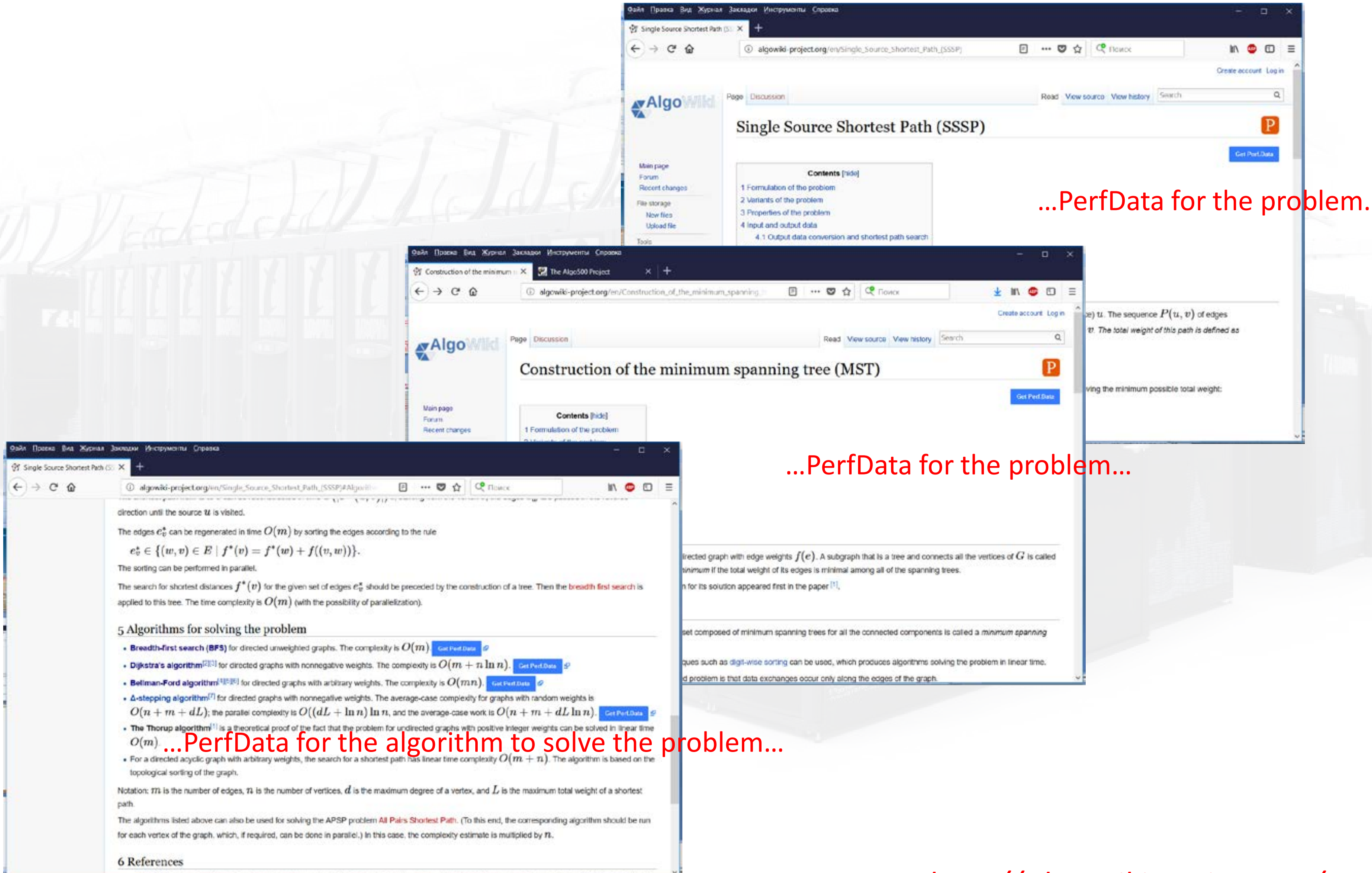
(+ run-time data: performance, time, efficiency… = PerfData)



How Does It Work ?

What Can We Obtain from AlgoWiki ?

Retrieve Performance Data from AlgoWiki...

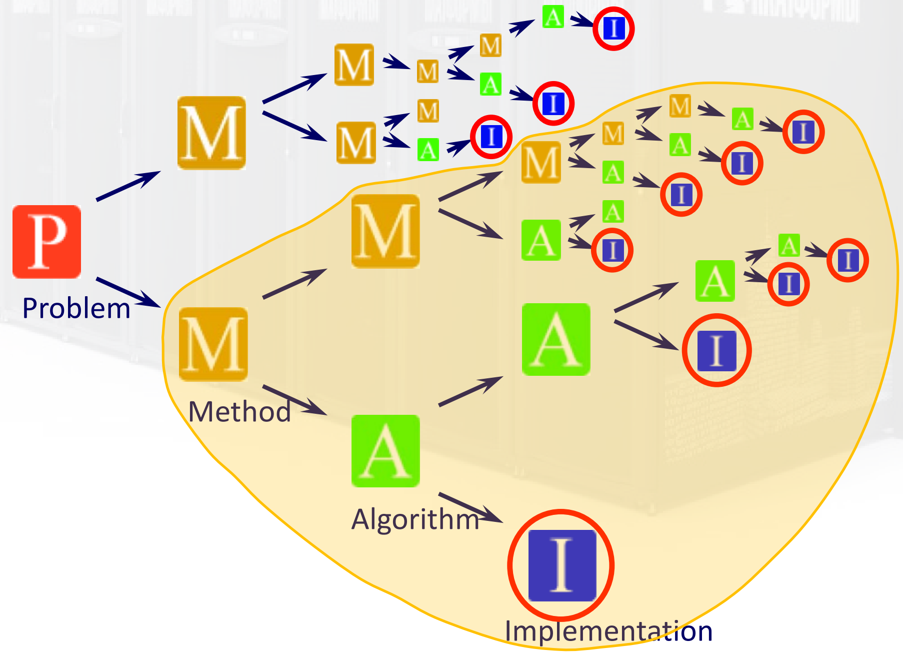


http://AlgoWiki-Project.org/en

“PerfData” list on “Strongly Connected Components”

(Method = Forward-Backward)

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



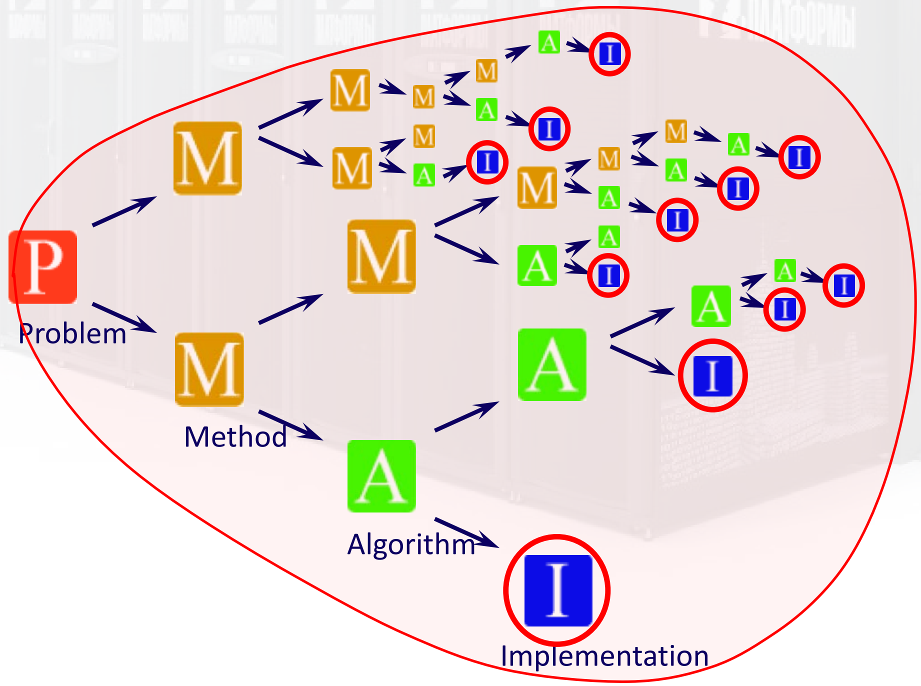
"PerfData" list on "Strongly Connected Components"
(Method = Forward-Backward)

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

“PerfData” list on “Strongly Connected Components”

(various methods, algorithms, implementations, computers)

- compare different ways of solving the problem;



"PerfData" 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

Data are submitted by different people…

What does it mean to submit performance data to AlgoWiki?

Path to the algorithm in AlgoWiki 1	Implementation details 2	Computer platform 3	Input data 4	Performance data 5

1. Information on the full chain: P → M → A → I → C (a path in AlgoWiki)

2. Source code or reference to a package. Compiler, compilation options and flags, libraries used, makefile, launch parameters...

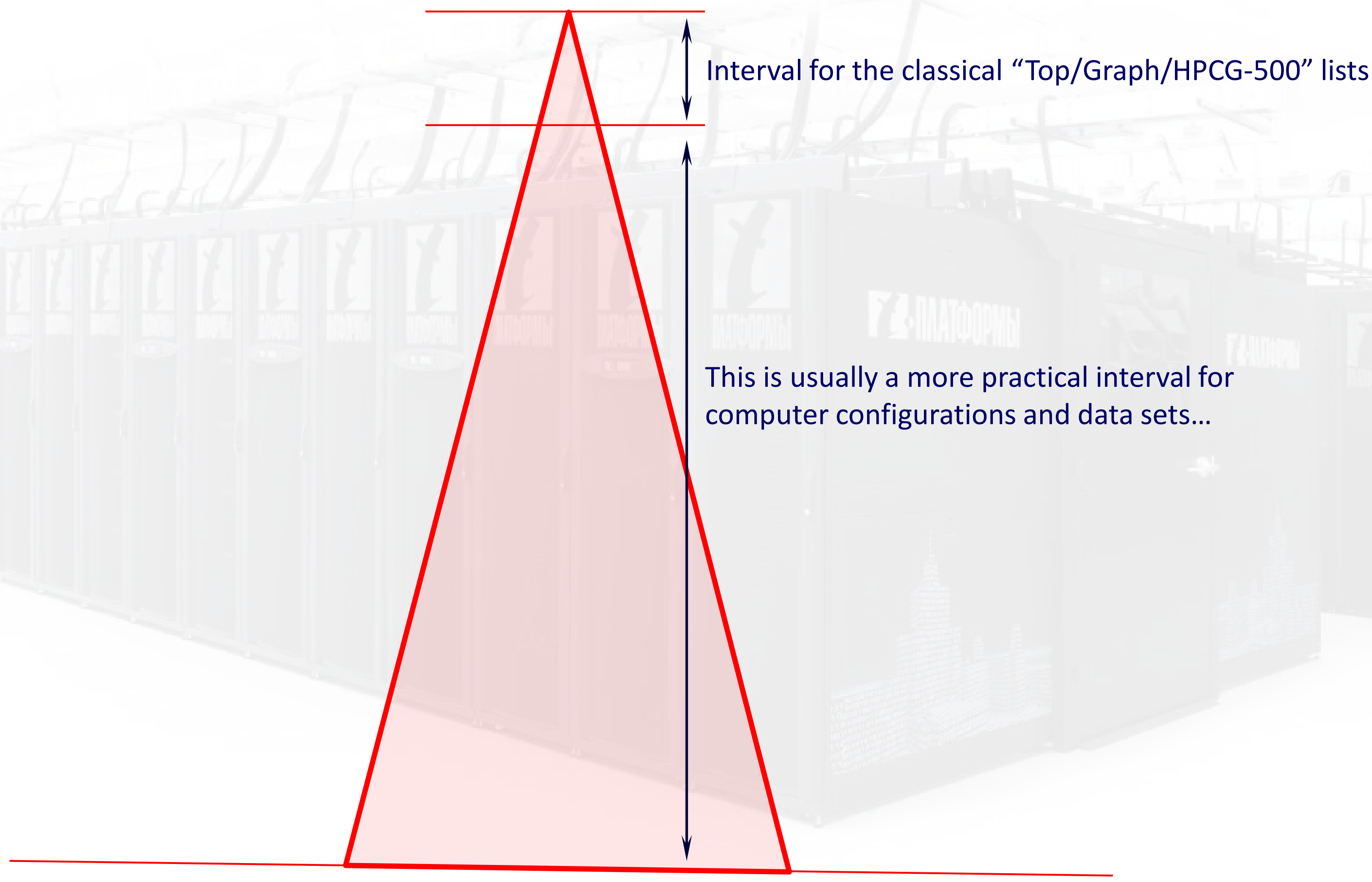
3. Configuration of the computing system with all necessary details.
   - Max / Medium / Small configurations.

4. Description of input data (matrices (dense, sparse, symmetric, …), graphs (RMAT, SSCA-2…), sizes, …)
   - Max / Medium / Small sizes.
   - Execution attributes (block size, processor matrix…)

5. Performance data:
   - Time, Flops, MTEPS, Gflops/Watt, efficiency (R_{max}/R_{peak}), …

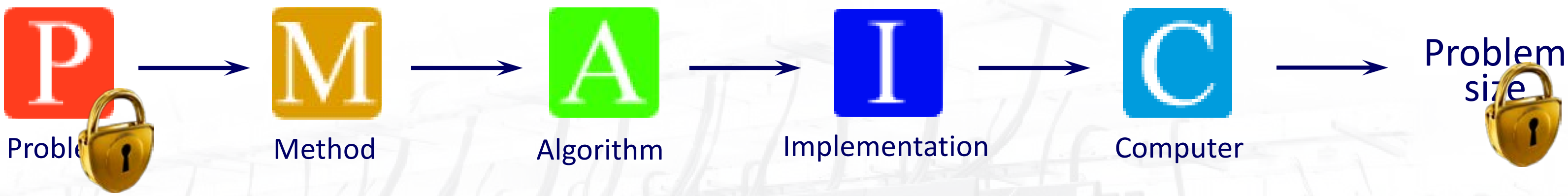
Important: we need to keep all data and details to be able **to reproduce** submitted performance results.

Max / Medium / Small – What does it mean?
(computer configurations, data sets)



“PerfData” list on “Single Source Shortest Path”

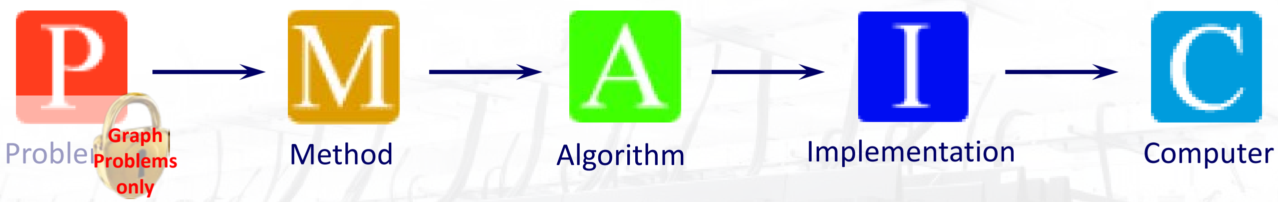
(How to solve the problem of the given size?)



Method	Implementation	Computing Platform	MTEPS	GraphType	GraphSize
Bellman–Ford	RCC for GPU	Lomonosov	1309,0	SSCA-2	2^20
Delta-Stepping	GAP	Lomonosov-2	512,0	RMAT	2^20
Bellman–Ford	Ligra	Lomonosov-2	511,0	RMAT	2^20
Bellman–Ford	RCC for GPU	Lomonosov	452,9	SSCA-2	2^20
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	204,1	RMAT	2^20
Bellman–Ford	RCC for CPU	Lomonosov	183,5	SSCA-2	2^20
Dijkstra's	PBGL MPI	Cluster / "Angara" interconnect	150,0	SSCA-2	2^20
Bellman–Ford	Graph500 MPI	Lomonosov	120,0	RMAT	2^20
Bellman–Ford	Graph500 MPI	Lomonosov	18,0	SSCA-2	2^20
Dijkstra's	PBGL MPI	IBM BlueGene/P	8,9	SSCA-2	2^20
Delta-Stepping	PBGL MPI	IBM BlueGene/P	3,8	SSCA-2	2^20
Delta-Stepping	PBGL MPI	IBM BlueGene/P	1,3	RMAT	2^20
Dijkstra's	PBGL MPI	IBM BlueGene/P	0,6	RMAT	2^20

"PerfData": the most efficient *graph applications*

(Where is this computer platform effective?)



Problem	Algorithm	Implementation	Platform	MTEPS	GraphType	GraphSize
Breadth-first search	BFS	RCC for GPU	Lomonosov-2 (NVIDIA P100)	11061	RMAT	2^22
Breadth-first search	BFS	GAP	Lomonosov-2 (NVIDIA K40)	5350	RMAT	2^22
Breadth-first search	BFS	Ligra	Lomonosov-2 (NVIDIA K40)	4168	RMAT	2^22
Minimum Spanning Tree	Boruvka's	RCC for GPU	Lomonosov-2 (NVIDIA P100)	3793	RMAT	2^26
Single Source Shortest Path	Bellman–Ford	RCC for GPU	Lomonosov (NVIDIA 2090)	1309,0	SSCA-2	2^20
Strongly Connected Components	Shiloach-Vishkin	Ligra	Lomonosov-2 (x86)	1307,00	RMAT	2^26
Single Source Shortest Path	Bellman–Ford	Ligra	Lomonosov-2 (x86)	1035,0	RMAT	2^21
Single Source Shortest Path	Delta-Stepping	PBGL MPI	Cluster / "Angara" net	809,5	SSCA-2	2^21
Page Rank	Page Rank	Nvidia nvGraph	Lomonosov-2 (NVIDIA K40)	753	RMAT	2^18
Breadth-first search	BFS	RCC for CPU	NEC SX-ACE	715	RMAT	2^22
Strongly Connected Components	Forward-Backward	RCC for GPU	Lomonosov-2 (NVIDIA P100)	620,00	RMAT	2^21
Single Source Shortest Path	Delta-Stepping	GAP	Lomonosov-2 (x86)	616,0	RMAT	2^21
Strongly Connected Components	Forward-Backward	RCC for CPU	Lomonosov-2 (x86)	564,00	RMAT	2^24
Strongly Connected Components	Shiloach-Vishkin	GAP	Lomonosov-2 (x86)	547,00	RMAT	2^20
Minimum Spanning Tree	Boruvka's	RCC for GPU	Lomonosov (NVIDIA 2090)	204,441	SSCA-2	2^14
Page Rank	Page Rank	GAP	Lomonosov-2 (NVIDIA K40)	172	RMAT	2^24
Breadth-first search	BFS	PBGL MPI	IBM BlueGene/P	167	SSCA-2	2^22
Breadth-first search	BFS	PBGL MPI	Lomonosov (x86)	155	SSCA-2	2^22
Minimum Spanning Tree	Boruvka's	RCC for CPU	Intel Xeon Phi (KNL)	25,16	SSCA-2	2^21
Single Source Shortest Path	Dijkstra's	PBGL MPI	IBM BlueGene/P	8,9	SSCA-2	2^20

AlgoWiki: an easy step back to analyze "PerfData"

(theoretical potential of algorithms is described in AlgoWiki)

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

P → M → A → I → C

Problem → Method → Algorithm → Implementation → Computer

AlgoWiki

Page | Discussion

Main page
Forum
Recent changes
File storage
New files
Upload file
Tools
What links here
Related changes
Special pages
Printable version

Dijkstra's algorithm

Primary authors of this description: A.N.Daryin, Vad.V.Voevodin (Section 2.2).

Contents [hide]

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 $O(n^2)$.

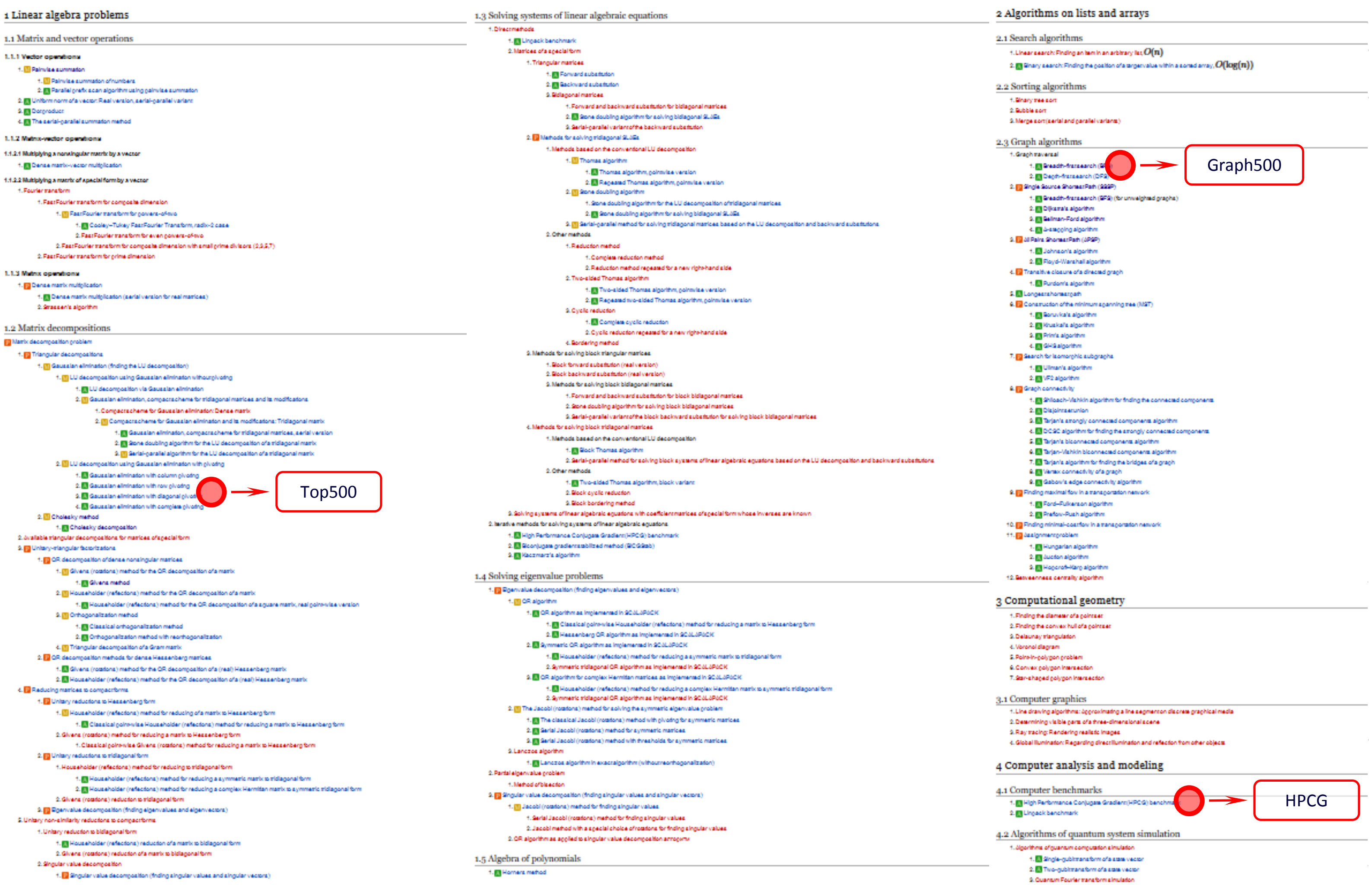
1.8 Parallelization resource of the algorithm

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

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

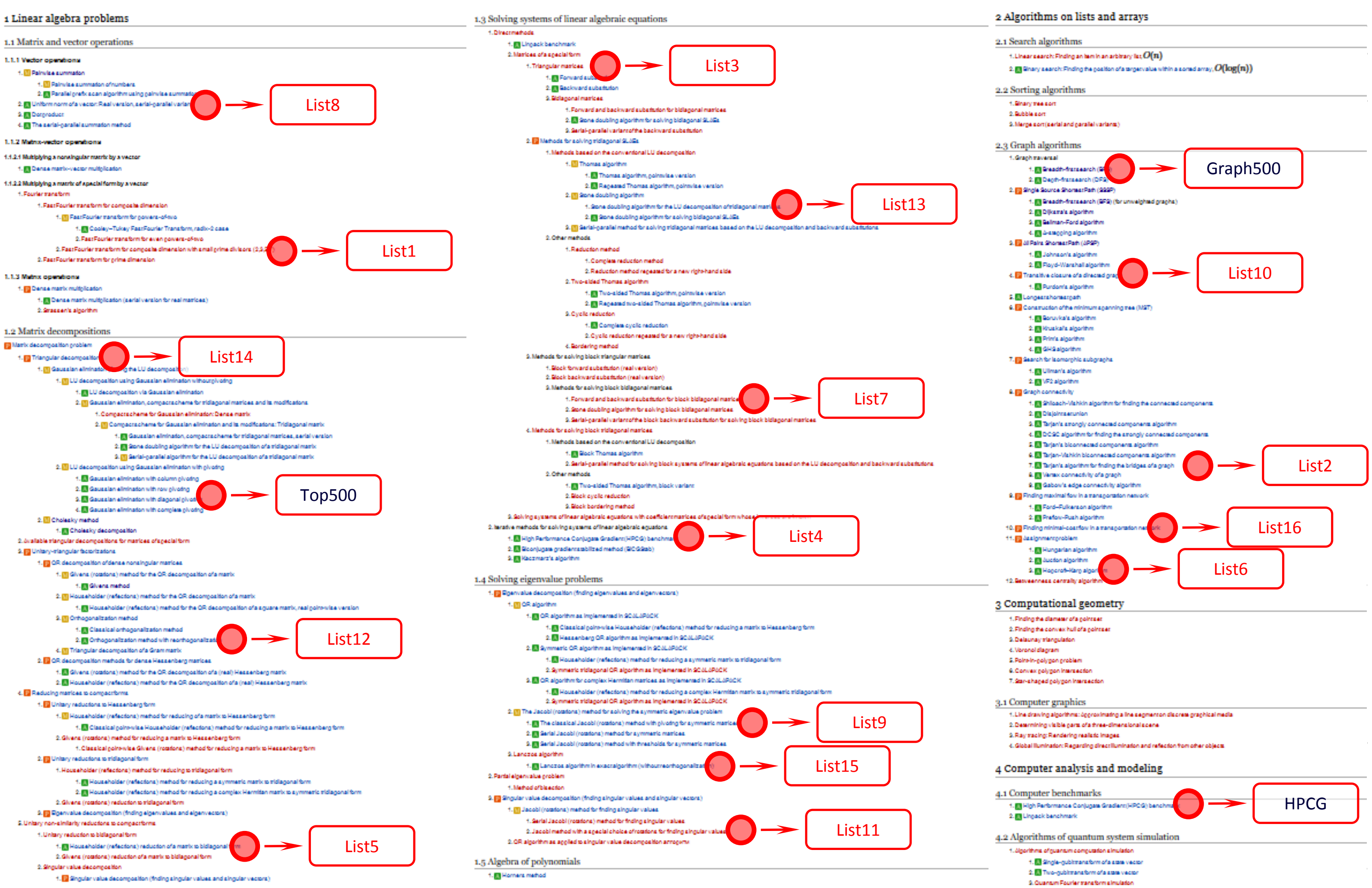
Algorithm classification and computer comparison

(What have we had up to now? Three points on the entire set of algorithms)



Algorithm classification and computer comparison

(What can AlgoWiki add to the standard three benchmarks/lists?)



General methodology to compare computing platforms

(using any algorithm)

problems for evaluation of computer platforms

AlgoWiki as an extension of the existing benchmarking methodology

Well-known theoretical potential of algorithms

Well-described and available community experience

AlgoWiki + Performance Data

Thank you !

AlgoWiki is the basis for
the Intelligible Parallel Computing World!

AlgoWiki-Project.org

September, 10th 2018, LIT JINR, Dubna