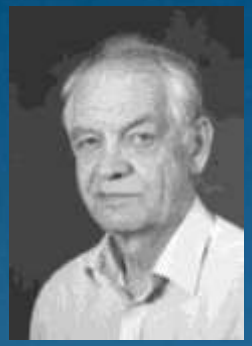


V.P.IVANNIKOV MEMORIAL WORKSHOP

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

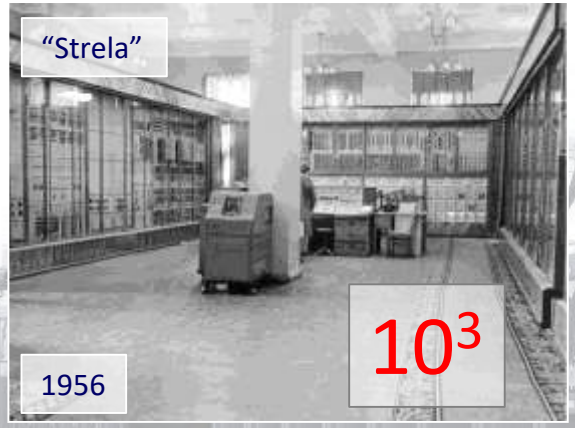
Corresponding Member of RAS

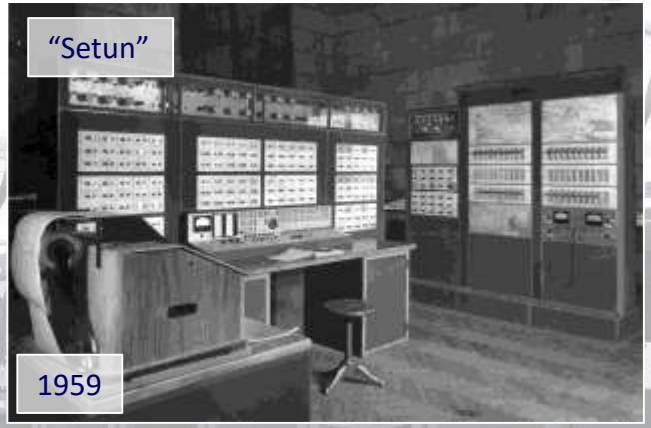
voevodin@parallel.ru

May, 3rd 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

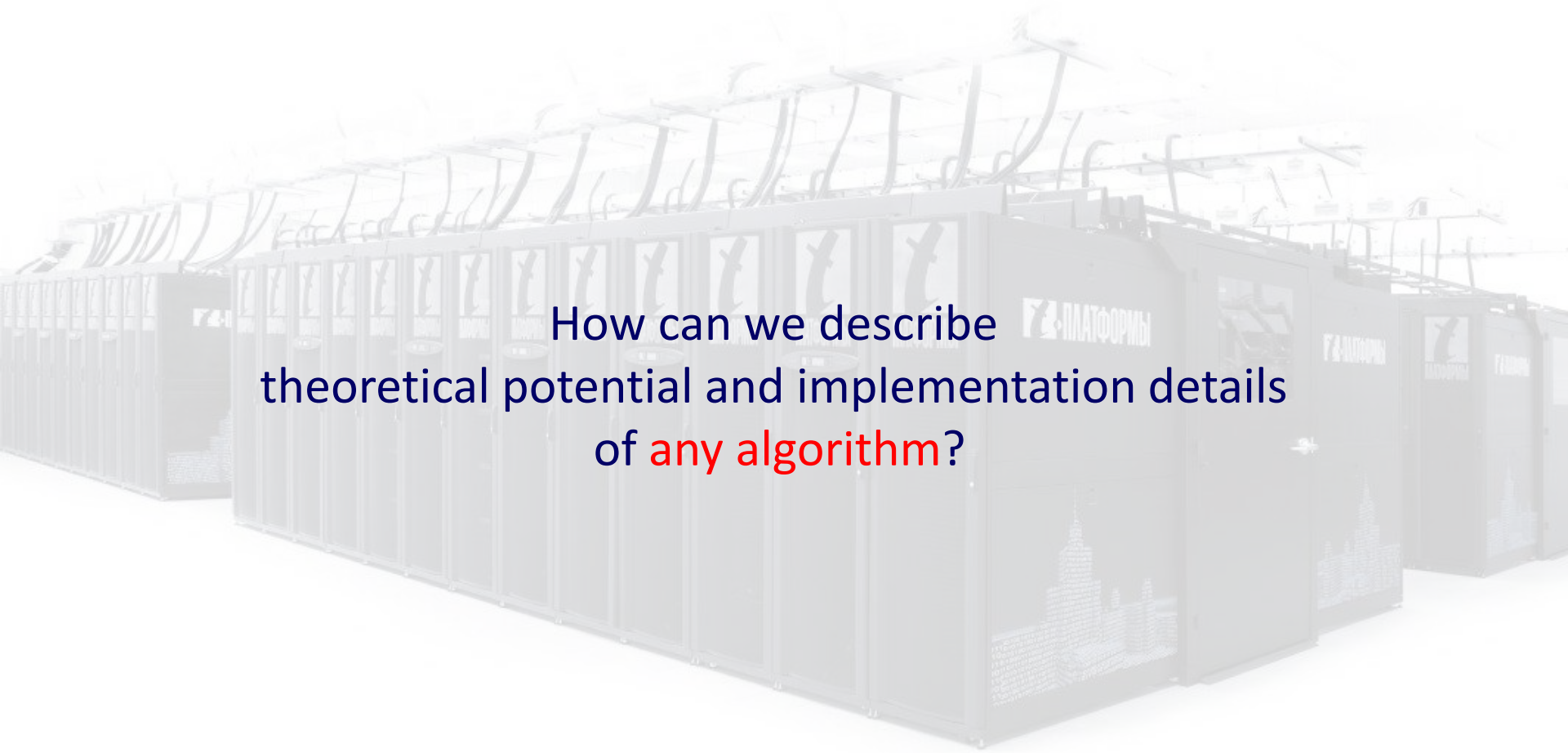
?

?

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
Macrostructure
Locality of computations
Scalability
Performance
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 mat

General Description

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

ed uniquely for the elements of the matrix L from

features.

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

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

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

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

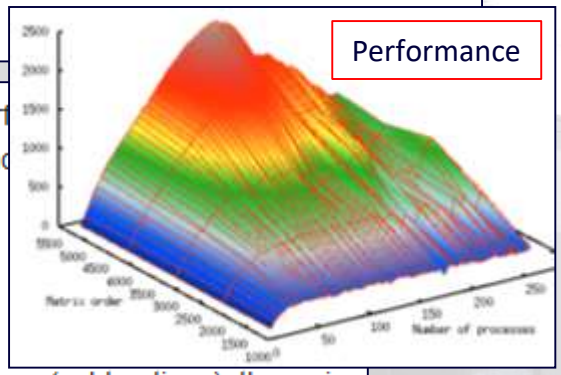
$$l_{ji} = \left(a_{ji} - \sum_{p=1}^{i-1} l_{ip} l_{jp}\right) / l_{ii}, \quad i \in [2, n-1], j \in [i+1, n].$$

The following number of operations should be perf

matrix of order n using a serial version of the Cho

Serial Complexity

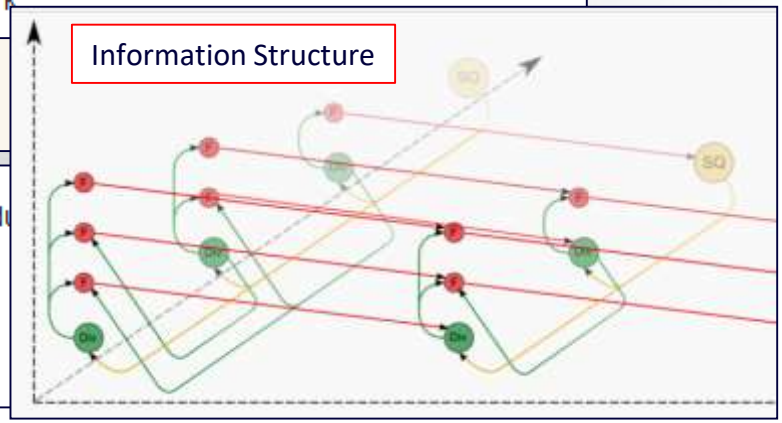
- n square roots,
- $\frac{n(n-1)}{2}$ divisiona,
- $\frac{n^3 - n}{6}$ multiplications and $\frac{n^3 - n}{6}$ additions (subtractions): the main amount of computational work



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



Description of Algorithms

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

Information Graph

Determinacy

Computational kernel

Locality of computations

Macrostructure

Performance

Scalability

Data locality

Mathematical description

Communication profile

Properties and Features

Efficiency

Serial Complexity

Computational intensity

Resource of Parallelism

Input / Output data

Algorithms: Theoretical Part

(machine-independent properties,

"Once and for all")

Algorithms: Implementation Issues

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

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.

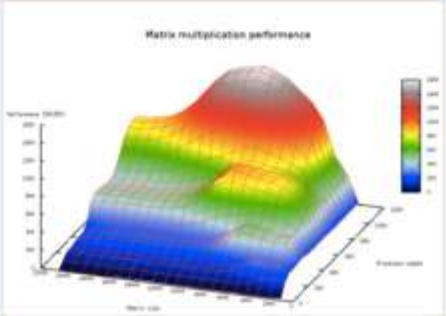
Properties of the algorithm:

- 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: $\frac{n(n+1)}{2}$

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

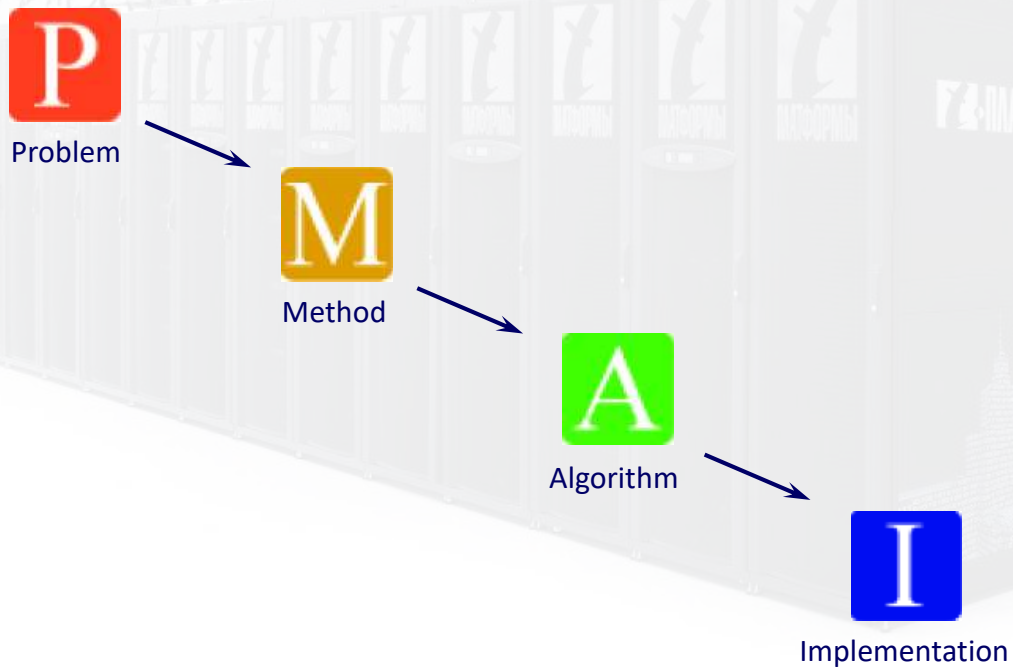
http://AlgoWiki-Project.org

AlgoWiki: algorithm classification
(tree structure)

Algorithm classification

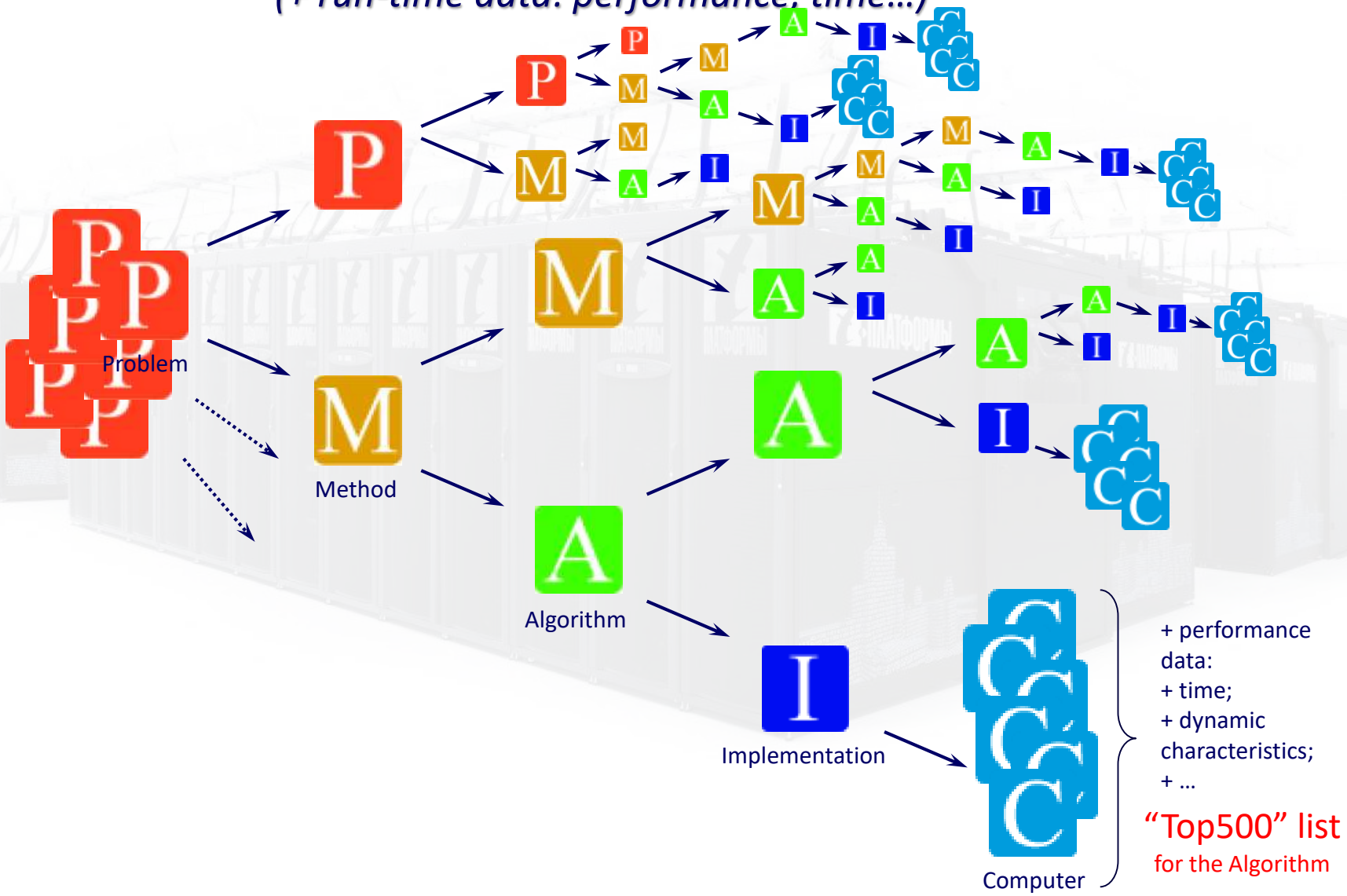
AlgoWiki: Problem – Method – Algorithm – Implementation

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



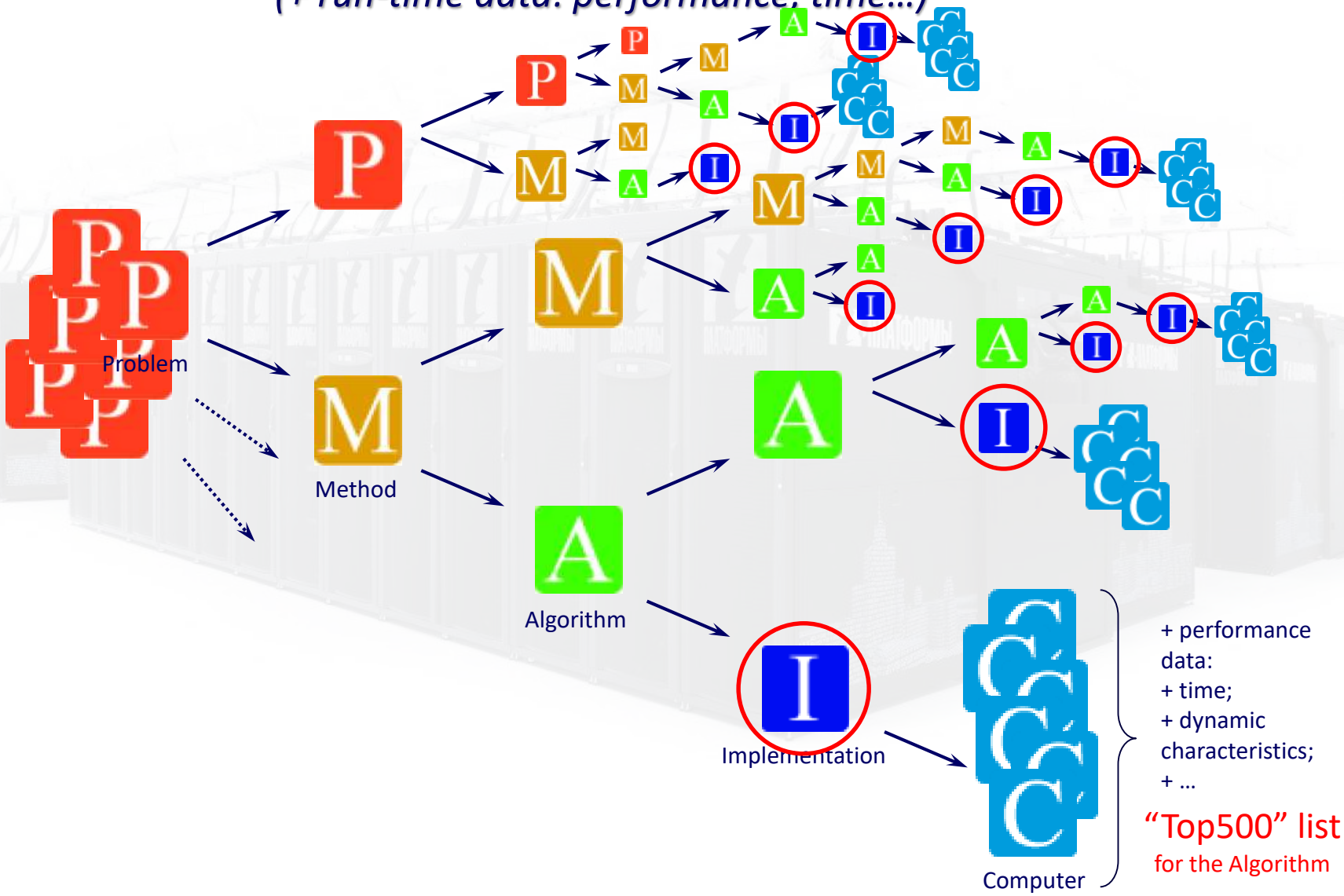
AlgoWiki: from Problems to various Implementations

(+ run-time data: performance, time…)



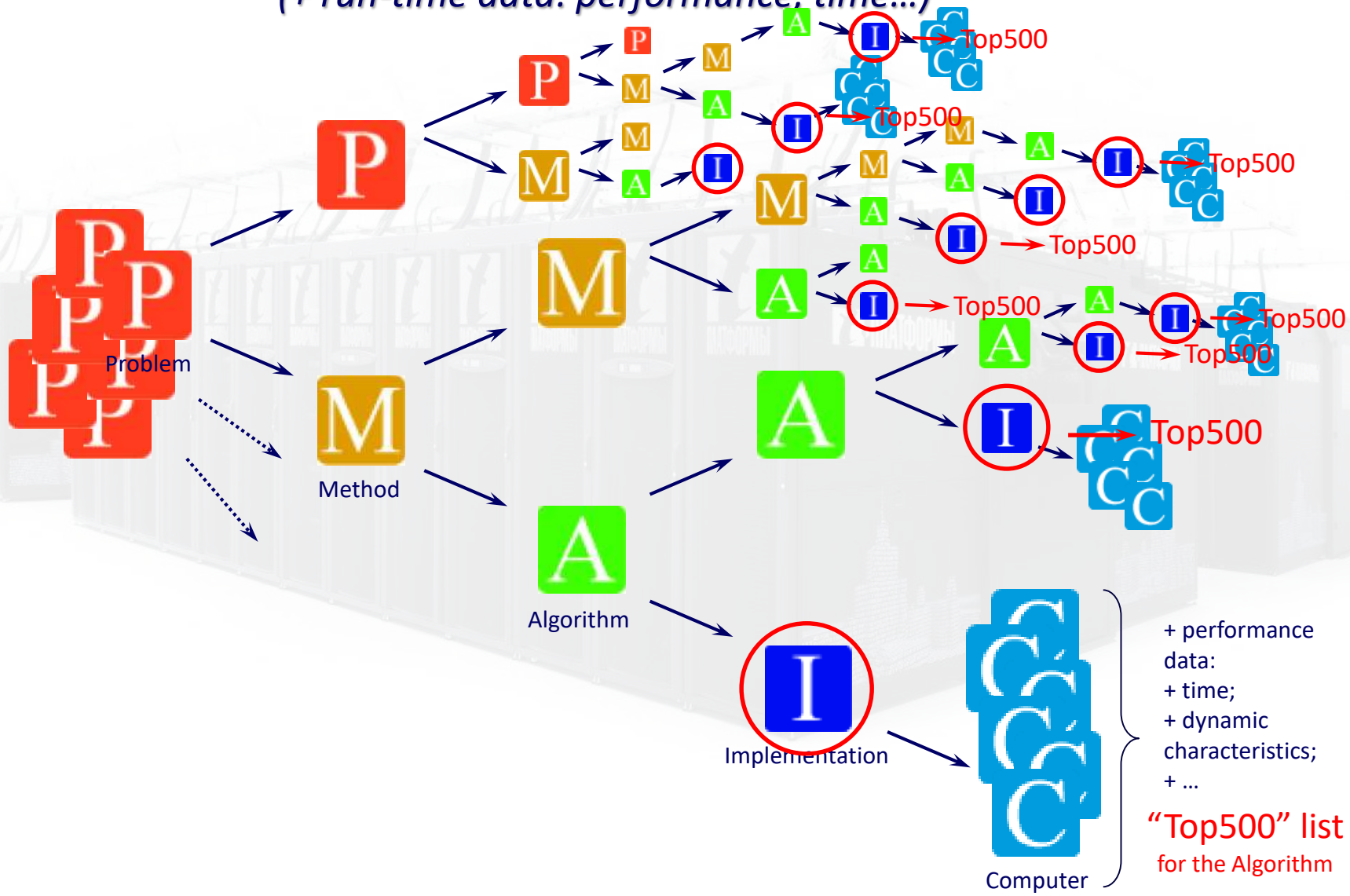
AlgoWiki: from Problems to various Implementations

(+ run-time data: performance, time…)



AlgoWiki: from Problems to various Implementations

(+ run-time data: performance, time…)

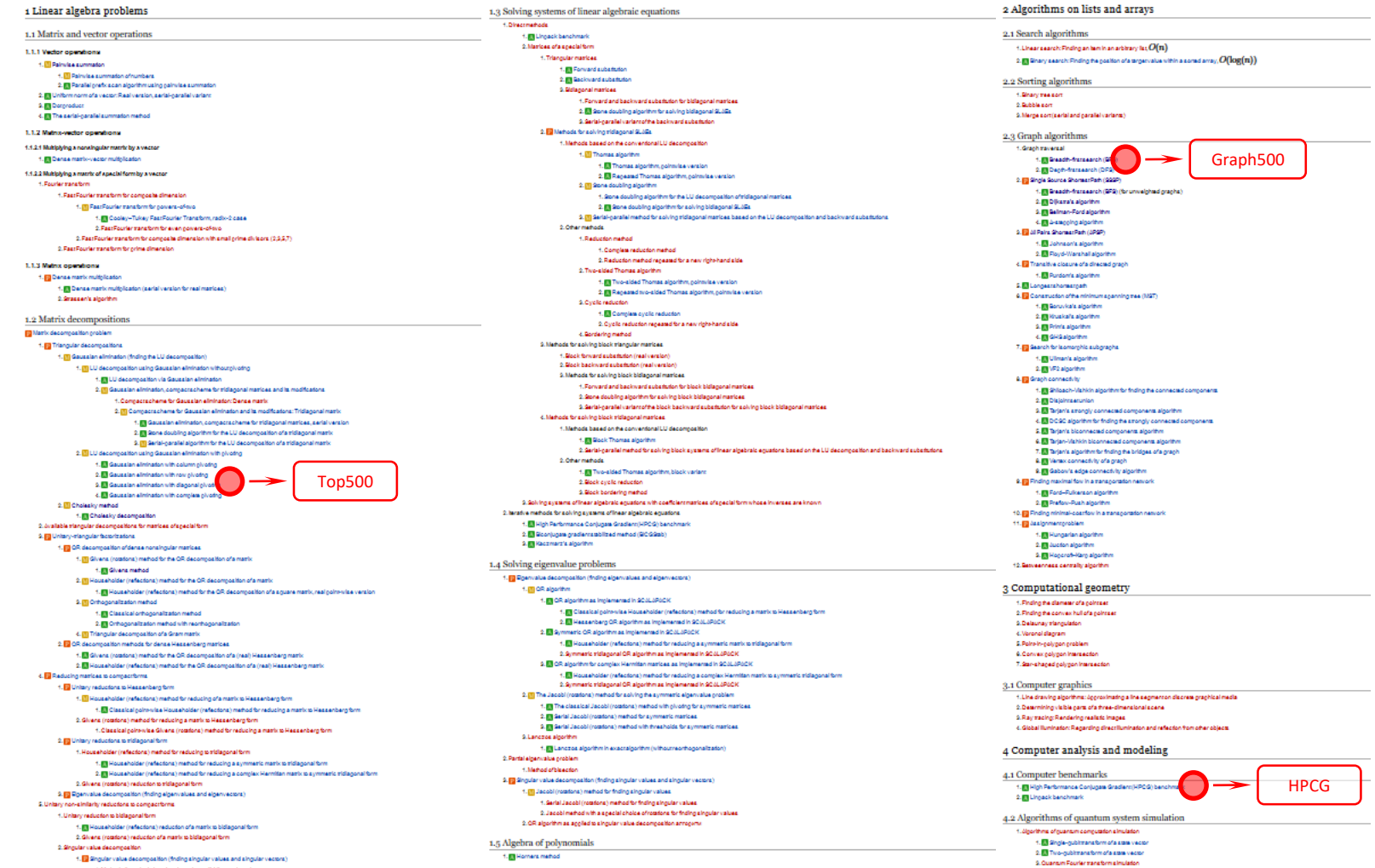


AlgoWiki

Algorithm classification and comparison of computers

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

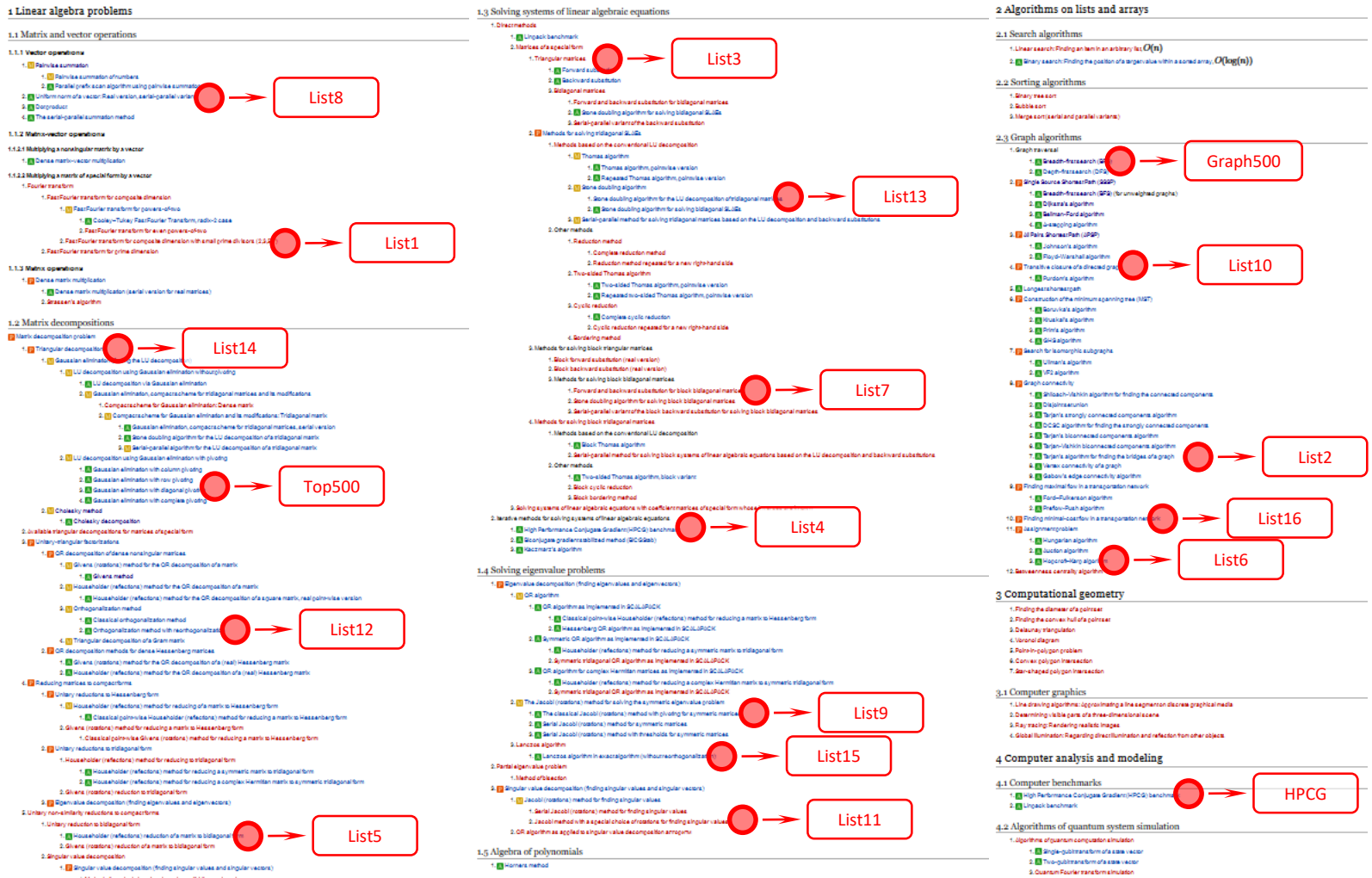
Algorithm classification



Algorithm classification and comparison of computers

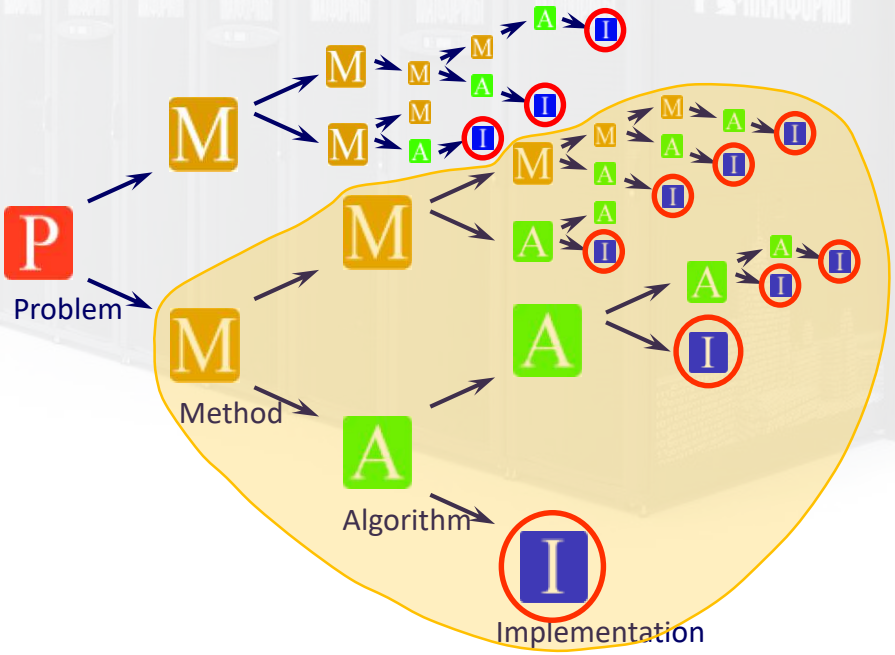
(what can AlgoWiki add to the standard three lists?)

Algorithm classification



"Top500" list for the given Problem and Method

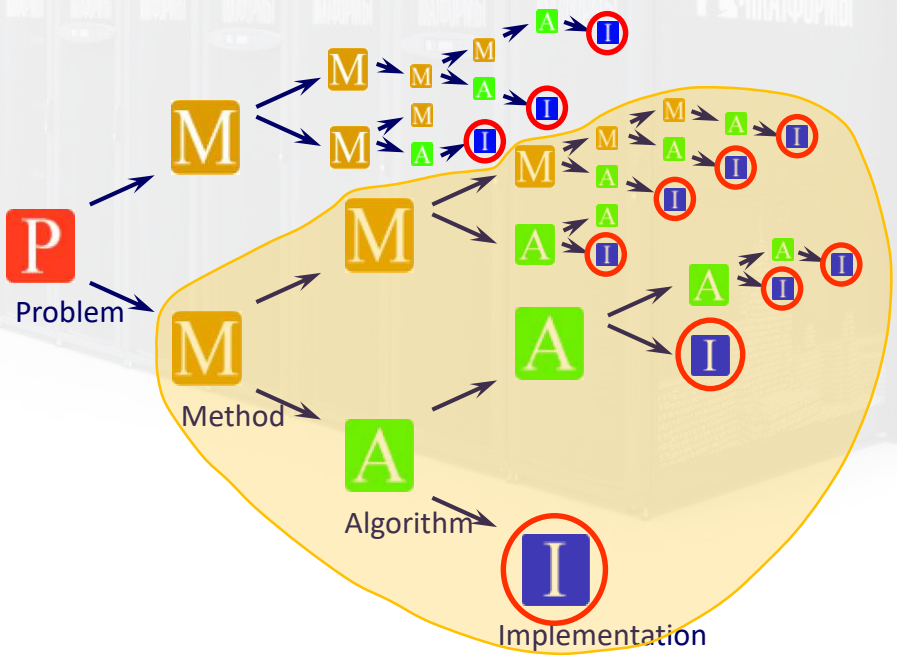
- compare different algorithms, implementations and computing platforms for the problem and method;



"Top500" list on "Strongly Connected Components"

(Method = Forward-Backward)

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



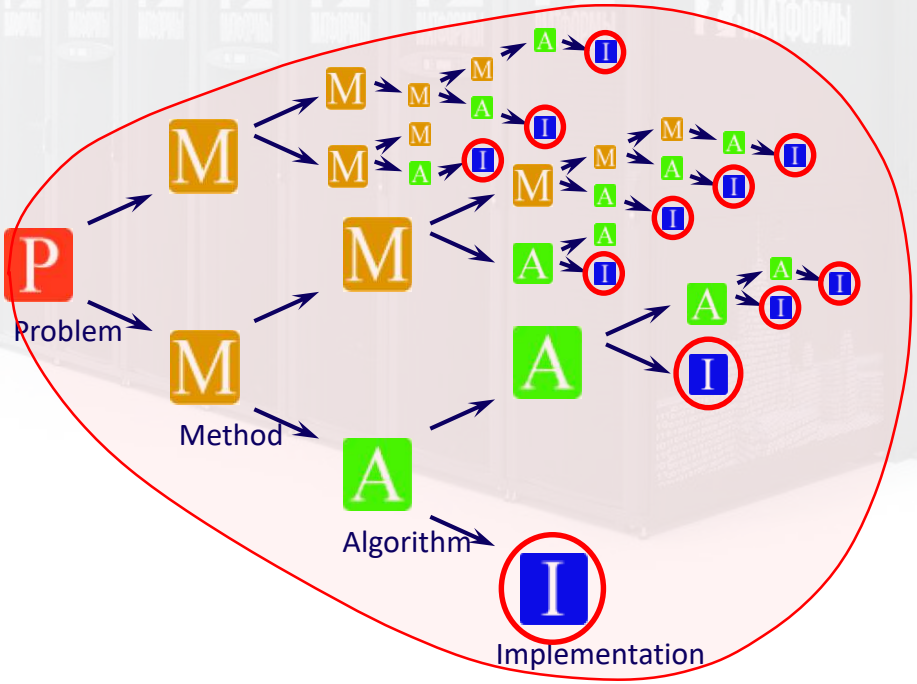
"Top500" list on "Strongly Connected Components"

(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

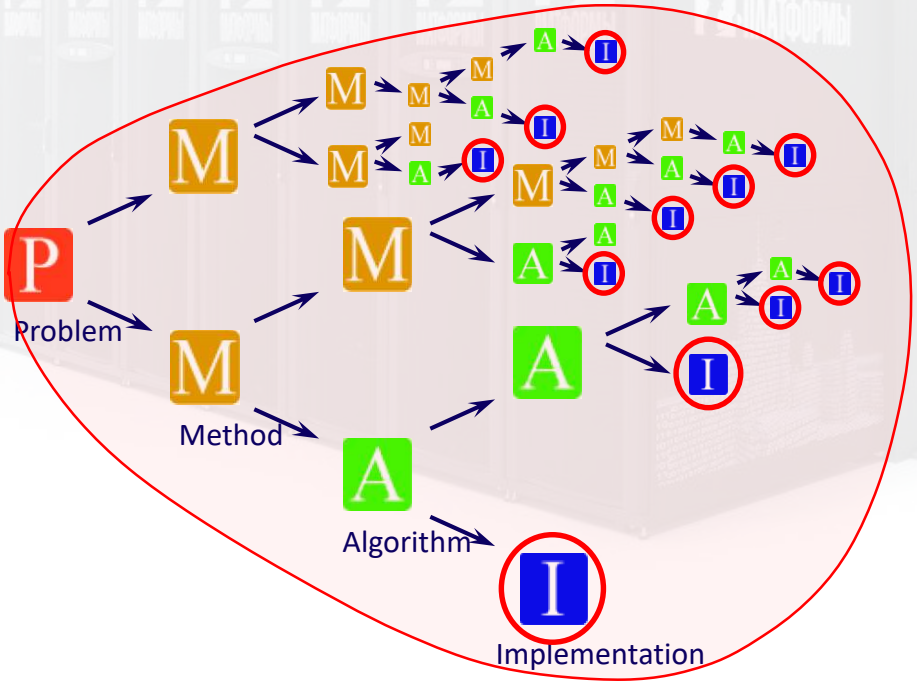
- compare different ways of solving the problem;



"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

P → M → A → I → C

Problem → Method → Algorithm → Implementation → Computer

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.

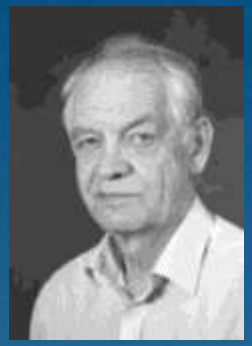
General methodology to compare computing platforms
(using any algorithm)

problems for evaluation of computer platforms

AlgoWiki as an extension of the Top500 Methodology ?

Well-known
theoretical potential

Well-described
community experience



Thank you !

AlgoWiki as an extension of the Top500 Methodology

Algowiki-Project.org

May, 3rd 2018, Erevan, Armenia