Lobachevsky State University of Nizhni Novgorod

Faculty of Calculating Mathematics and Cybernatics
Software Department

Parallel algorithm of graph ordering for minimizing sparse Cholesky factor fill-in

Ph.D. student
Pirova Anna

Moscow, 5 March 2015

Agenda

- Cholesky factorization
- Problem statement
- Reordering methods
- Nested dissection method
- Reordering method in MORSy
- Parallel algorithm for shared memory systems
- Experimental results
- Conclusions and future works

Cholesky factorization (1)

- System of linear equations

$$Ax = b \quad (1)$$

A – sparse symmetric positive definite matrix,
b – dense vector, x – vector of unknowns.

- Direct method – Cholesky factorization:

$$A = LL^T \quad (2)$$

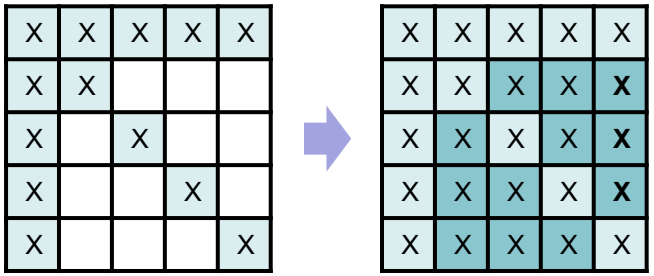
L – lower triangular matrix called *factor* of the matrix A

- Solve two triangular systems

$$Ly = b, L^T x = y \quad (3)$$

Cholesky factorization (2)

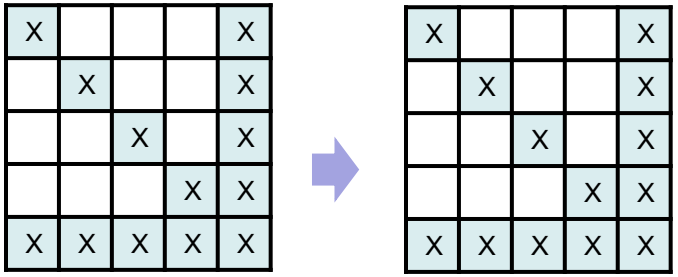
- Factor L *fill-in:*



- Matrix rows and columns reordering:

$$(PAP^T)(Px) = Pb$$

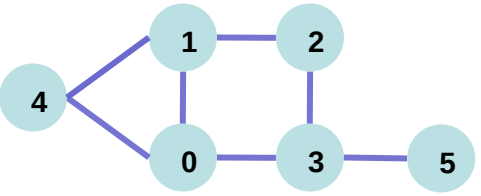
P – permutation matrix



Problem statement (1)

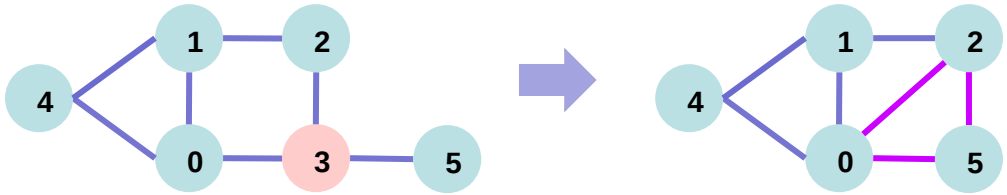
- $A = (a_{ij})$ sparse symmetric $n \times n$ matrix
- Graph $G = (V, E)$:
  - each vertex $v_i \in V$ is associated with row i.
  - each edge $(v_i, v_j) \in E$ is associated with $a_{ij} \neq 0$, $i \neq j$.

	0	1	2	3	4	5
0	9	1			2	
1	1	8	3		2	
2		3	9	4		1
3			4	7		
4	2	2			5	
5			1			6



Problem statement (2)

- Vertex v *elimination*:
  - v is removed with its edges
  - vertexes adjacent to v are collapsed to the clique



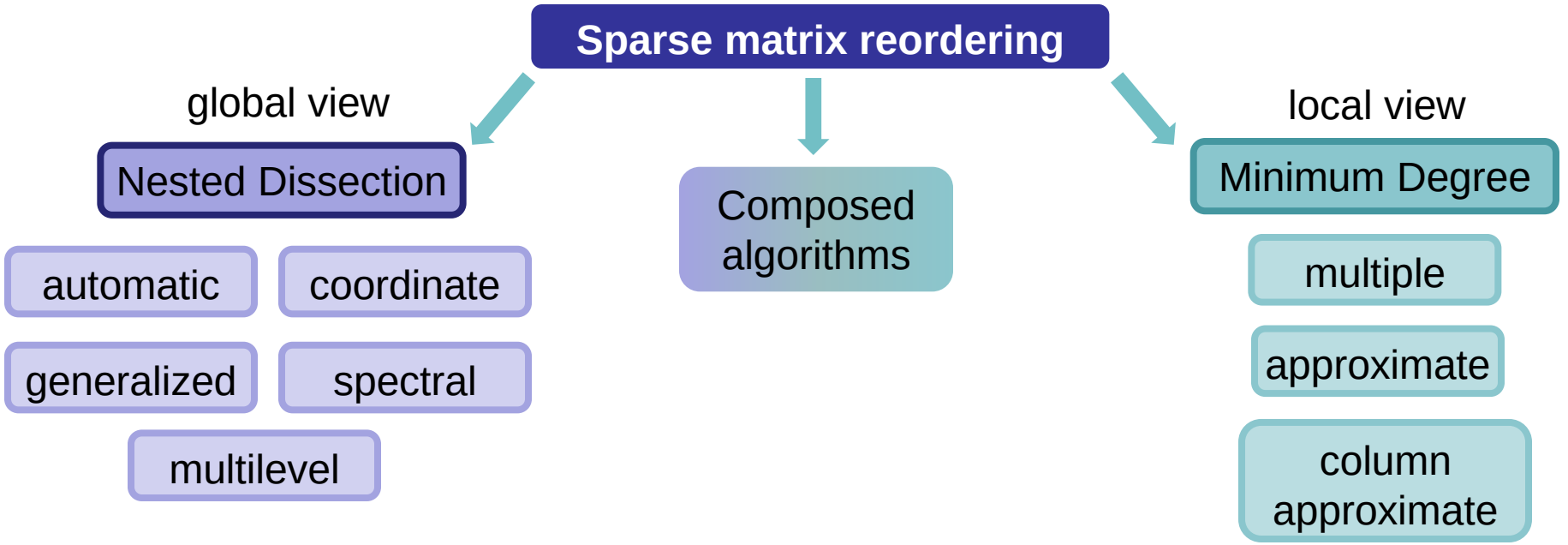
Problem statement (3)

- $\pi = (\pi_1, \pi_2, \ldots, \pi_n) \in \{\{1, 2, \ldots n\} \to \{1, 2, \ldots, n\}\}$ – permutation or *ordering* of V.
- *Fill-in* $F(\pi)$ generated by π – a set of edges added during consequent elimination of vertexes $\pi_1, \pi_2, \ldots, \pi_n$.
- Problem: finding ordering π^* that minimizes $|F(\pi^*)|$:

$$\pi^* = \underset{\pi \in \{\{1,2,\ldots n\} \to \{1,2,\ldots,n\}\}}{\operatorname{argmin}} |F(\pi)|$$

- Problem is NP-complete (*Yannakakis*, 1981).

Sparse matrix reordering methods

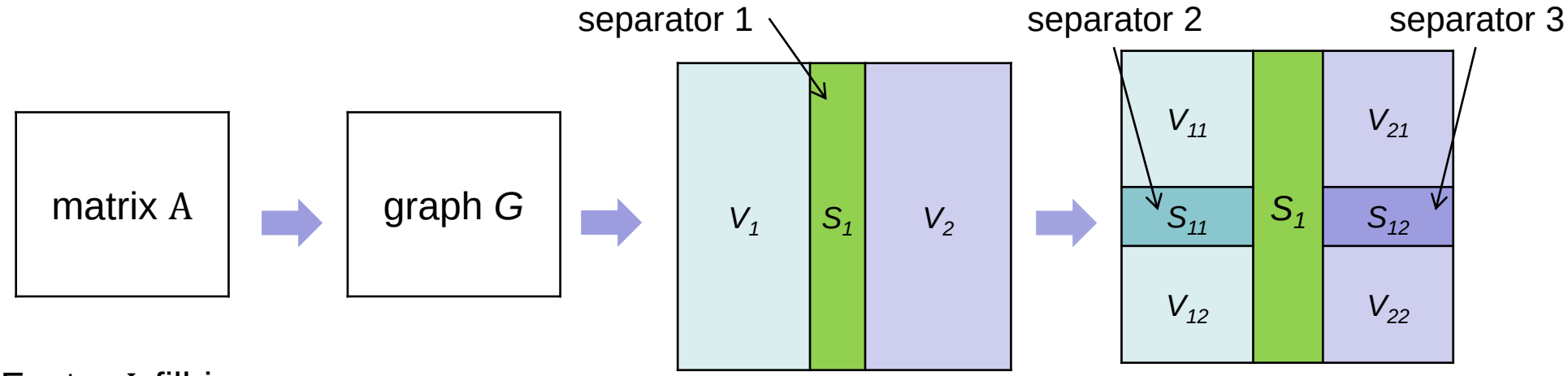


Software:

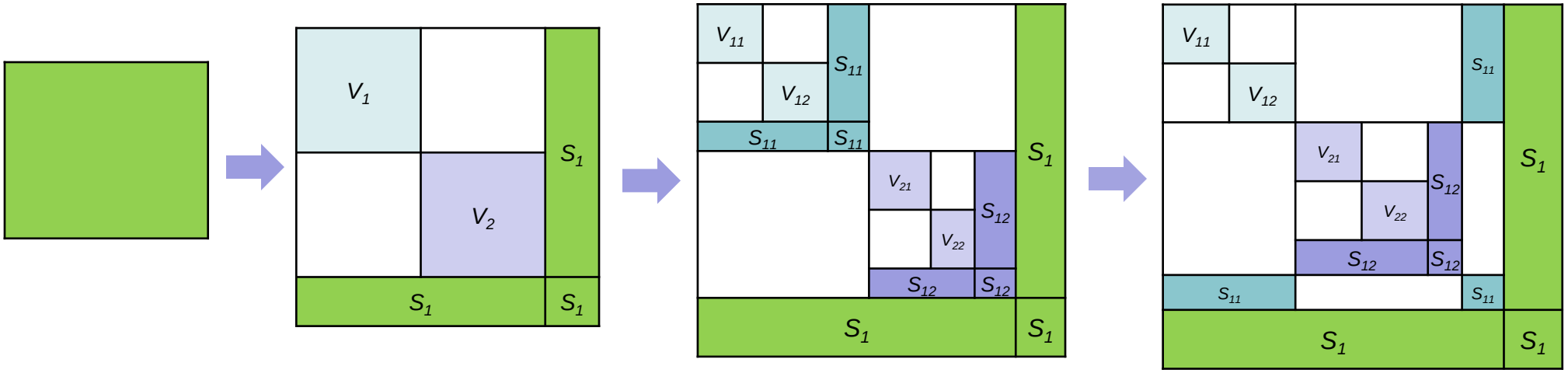
- METIS, Scotch, Chaco…;
- internal in sparse direct solvers (MUMPS, MKL, SuperLU…)

Parallel tools (distributed memory systems): ParMETIS, PT-Scotch

Nested dissection



Factor L fill-in:



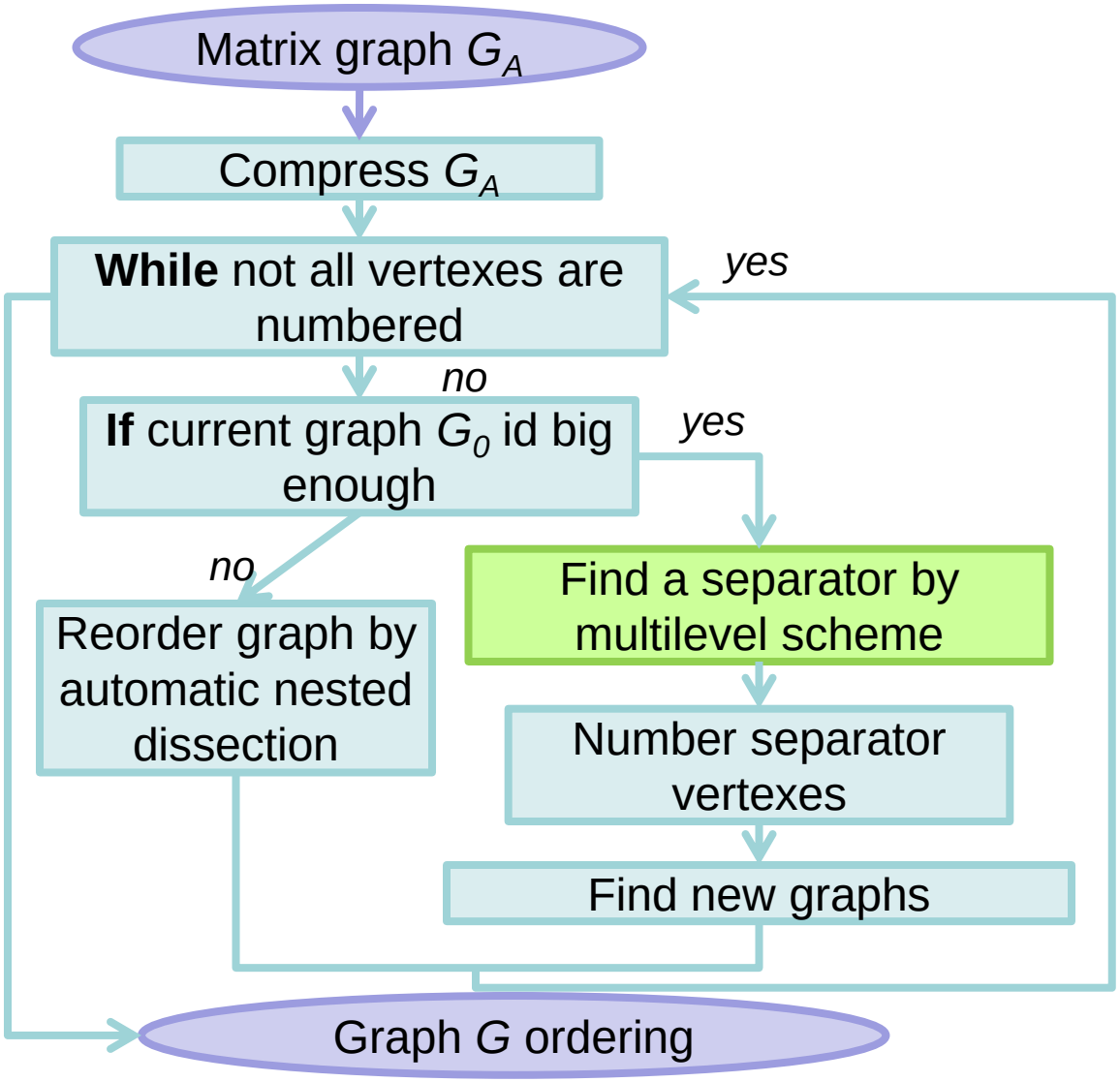
Modifications: separator calculating

MORSy

- MORSy – a new library for sparse matrix reordering to reduce its fill-in
  - publically available from

    http://hpc-education.unn.ru/en/research/overview/sparse-algebra/morsy
  - written in C
  - cross-platform (Linux, Windows OS)
- MORSy is based on the multilevel nested dissection algorithm with modifications for vertex separator

MORSy reordering algorithm



Parallel algorithm for shared-memory systems. Task-based parallelism with concurrent queue (1)

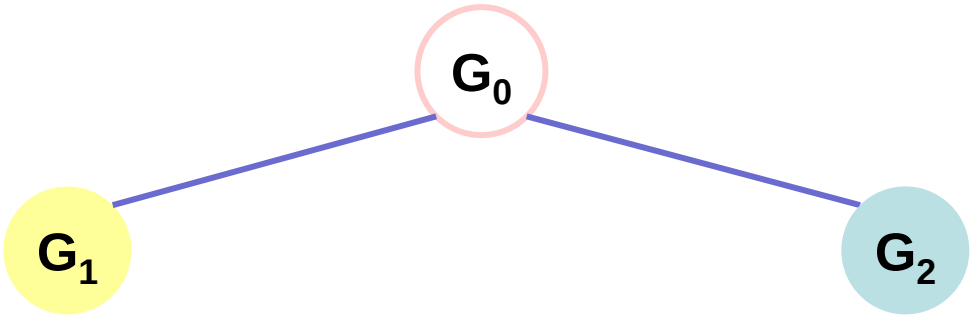
- OpenMP technology
- Parallel algorithm:
  - A computational task = to obtain a separator for the given subgraph
  - A shared concurrent queue is used for saving input and output data for computational tasks
  - A thread dequeues the first free task, obtains a separator for the associated graph, and enqueues the new subgraphs raised **after** graph separation
  - Parallel processing stops when separators for all graphs are found and there are no free tasks

Parallel algorithm for shared-memory systems. Task-based parallelism with concurrent queue (2)

The queue

Parallel algorithm for shared-memory systems. Task-based parallelism with concurrent queue (3)

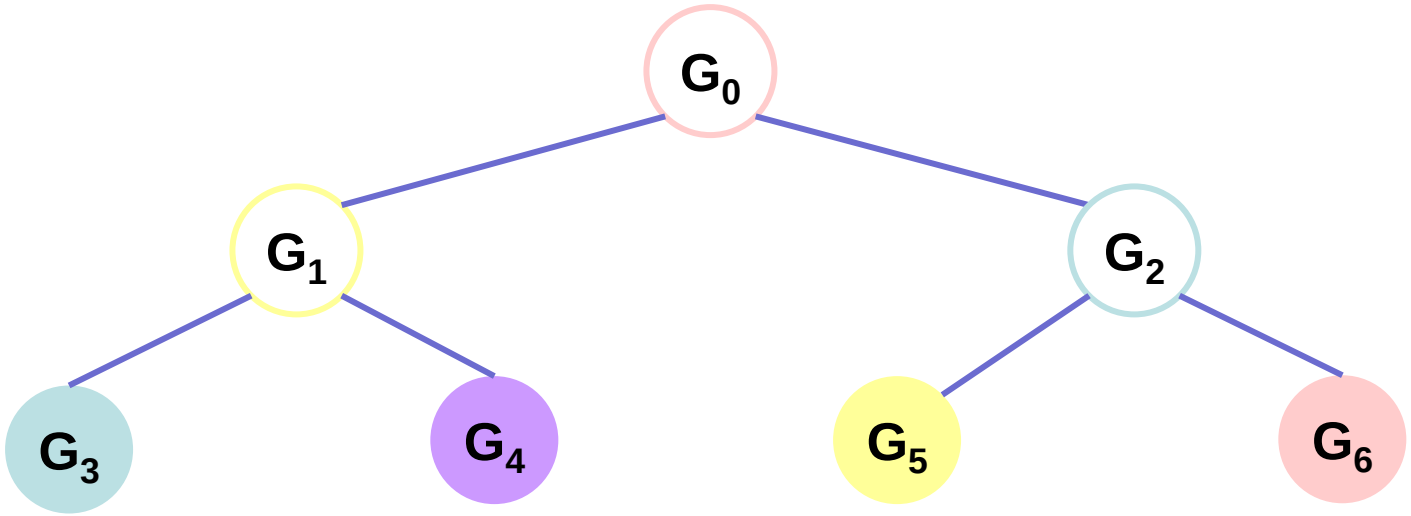
The queue



Parallel algorithm for shared-memory systems. Task-based parallelism with concurrent queue (4)

The queue

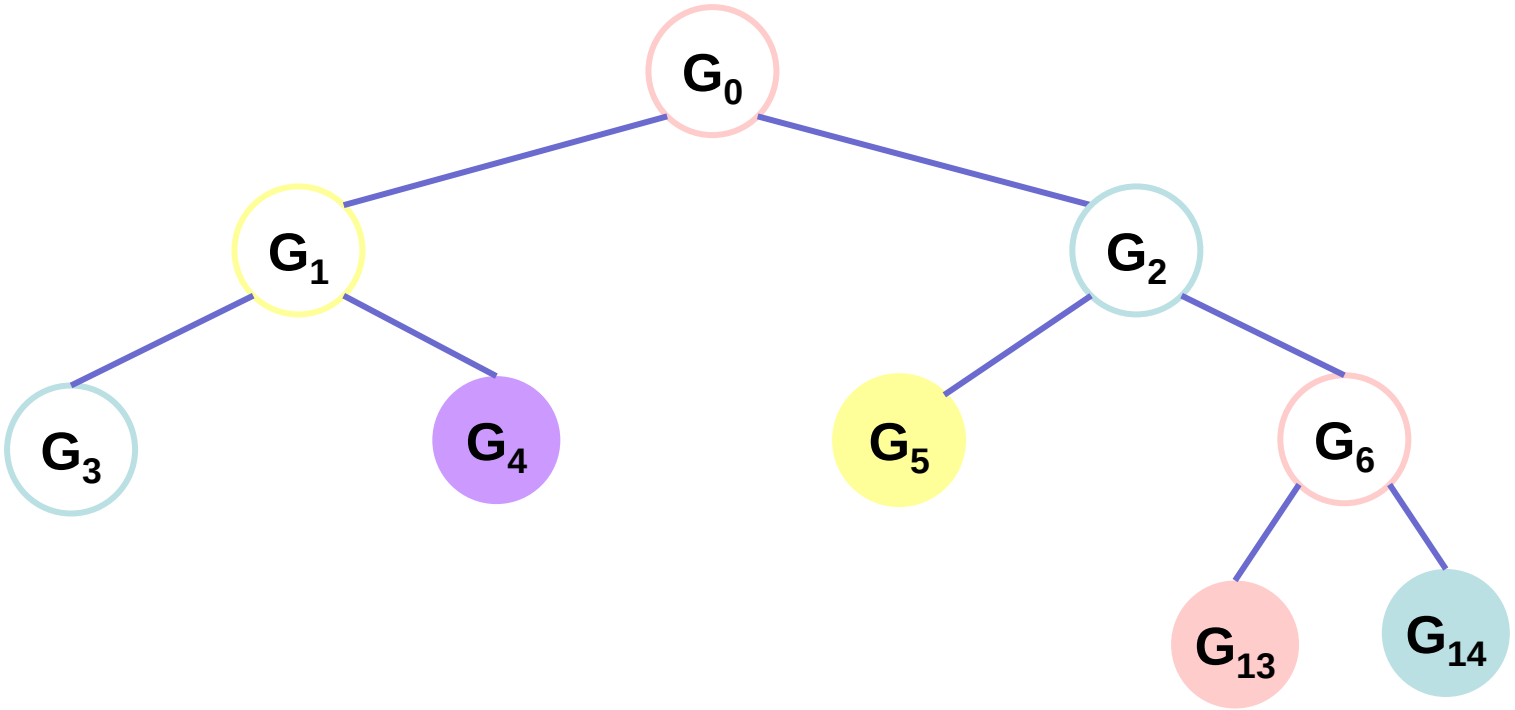
G_0	G_1	G_2	G_5	G_6	G_3	G_4								



Parallel algorithm for shared-memory systems. Task-based parallelism with concurrent queue (5)

The queue

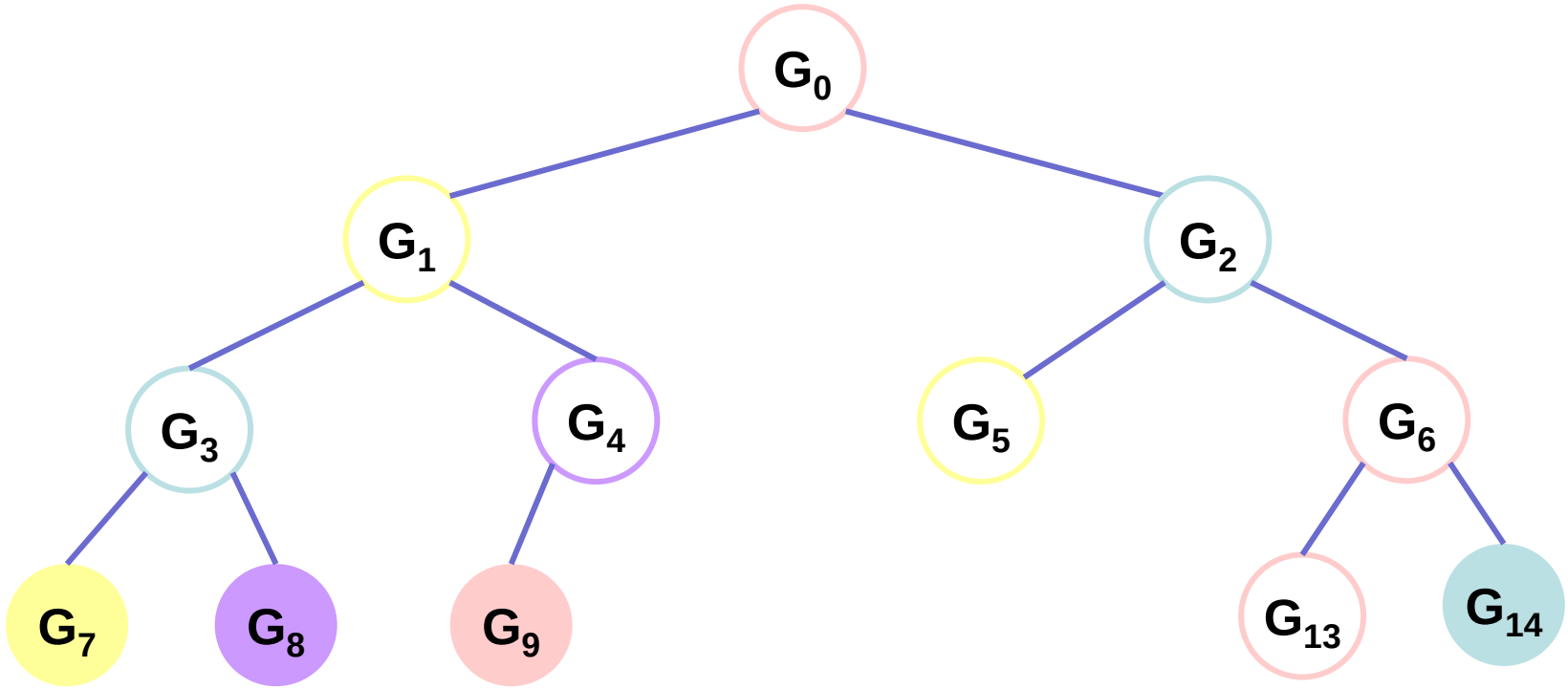
G_0	G_1	G_2	G_5	G_6	G_3	G_4	G_{13}	G_{14}	G_7	G_8					



Parallel algorithm for shared-memory systems. Task-based parallelism with concurrent queue (6)

Очередь задач

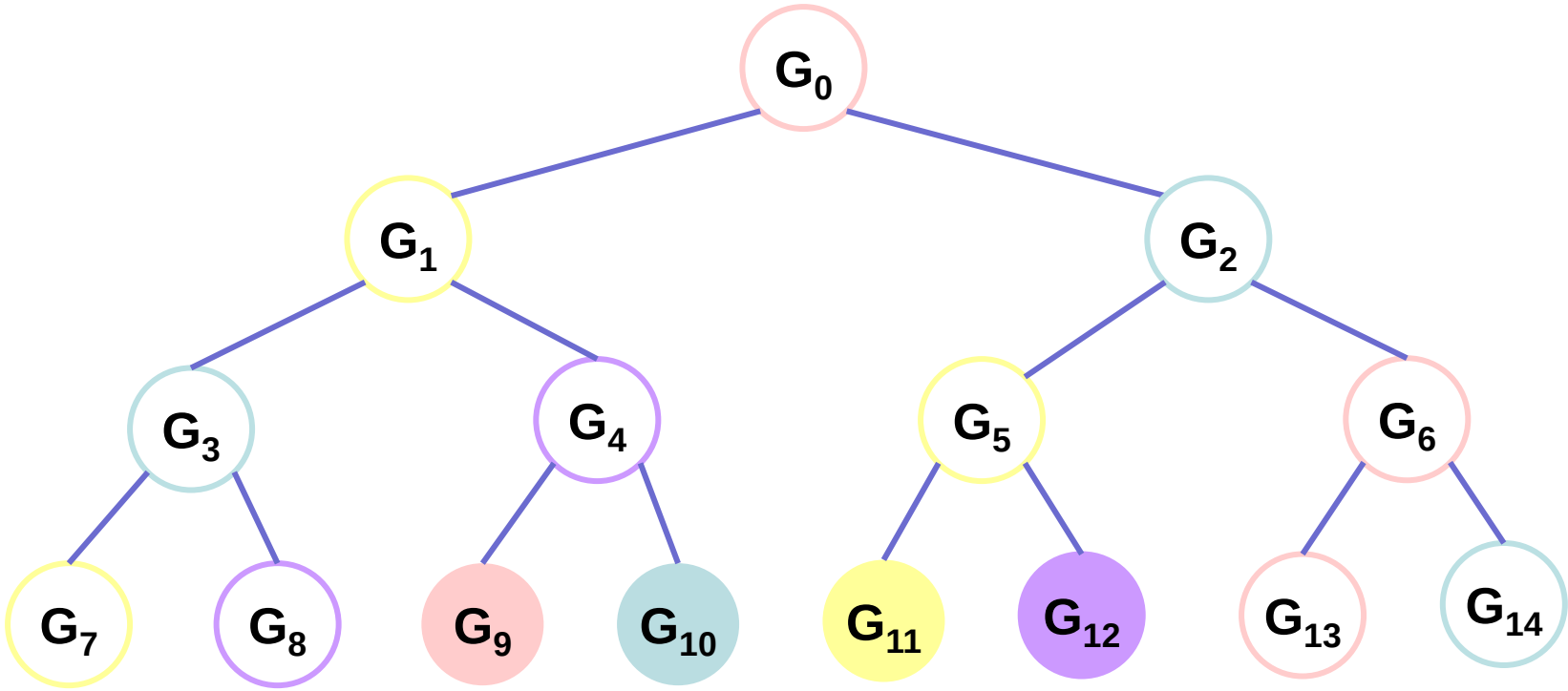
G_0	G_1	G_2	G_5	G_6	G_3	G_4	G_{13}	G_{14}	G_7	G_8	G_9	G_{10}	G_{11}	G_{12}



Parallel algorithm for shared-memory systems. Task-based parallelism with concurrent queue (7)

Очередь задач

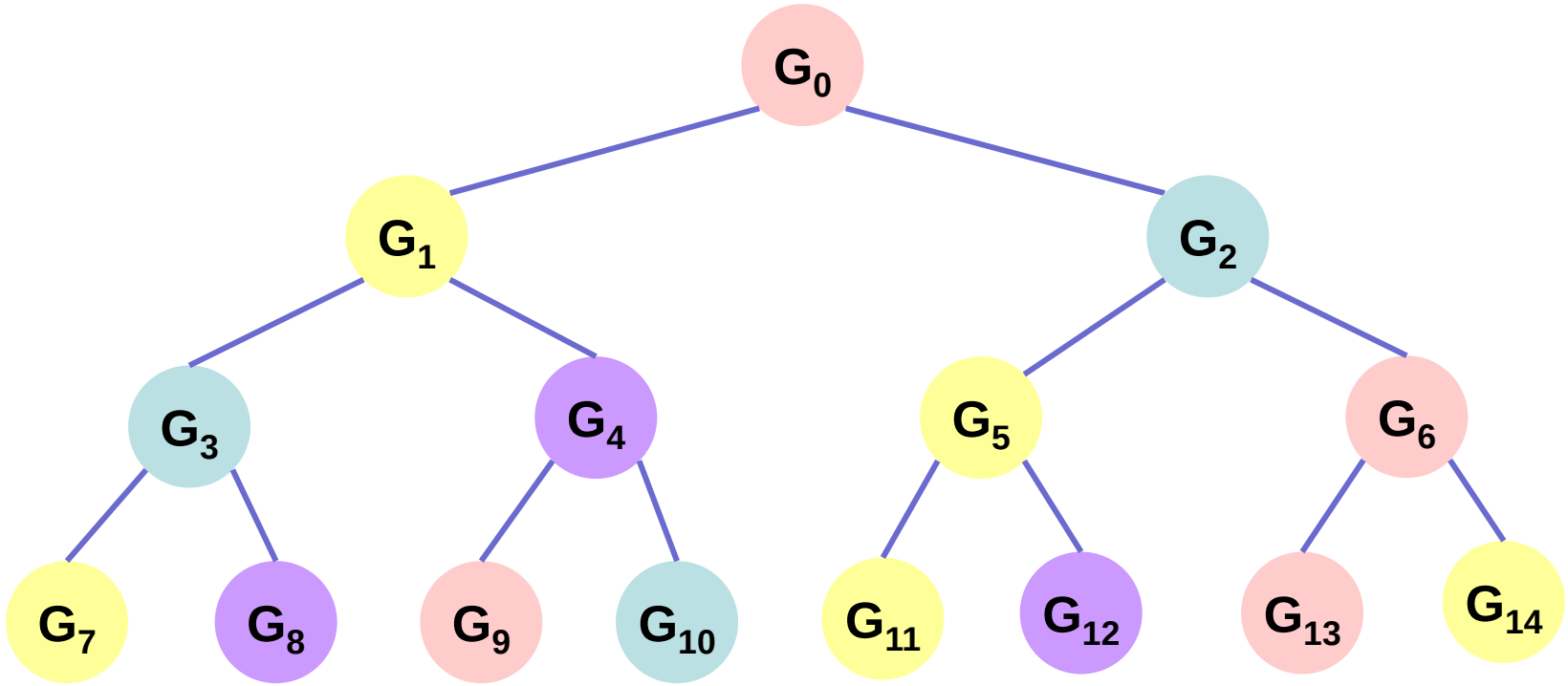
G_0	G_1	G_2	G_5	G_6	G_3	G_4	G_{13}	G_{14}	G_7	G_8	G_9	G_{10}	G_{11}	G_{12}



Parallel algorithm for shared-memory systems. Task-based parallelism with concurrent queue (8)

The queue

G_0	G_1	G_2	G_5	G_6	G_3	G_4	G_{13}	G_{14}	G_7	G_8	G_9	G_{10}	G_{11}	G_{12}



Parallel algorithm for shared-memory systems. Task-based parallelism with OpenMP tasks

- A computational task = to obtain a separator for the given subgraph
- Recursive algorithm of reordering
- The task starts by directive #pragma omp task

Testing (1)

- Matrices from The University of Florida Sparse Matrix Collection (http://www.cise.ufl.edu/research/sparse/matrices/):

Name	Size, N	Number of non-zeros, NZ	Description
pwtk	217 918	11 524 432	structural problem
msdoor	415 863	19 173 163	structural problem
parabolic_fem	525 825	3 674 625	fluid dynamics problem
tmt_sym	726 713	5 080 961	electromagnetics problem
boneS10	914 898	40 878 708	3D structural problem
Emilia_923	923 136	40 373 538	3D structural problem
audikw_1	943 695	77 651 847	3D problem
bone010	986 703	47 851 783	3D problem
ecology2	999 999	4 995 991	2D structural problem
thermal2	1 228 045	8 580 313	unstructured FEM
StocF-1465	1 465 137	21 005 389	fluid dynamics problem
Hook_1498	1 498 023	59 374 451	3D structural problem
Flan_1565	1 564 794	114 165 372	3D structural problem

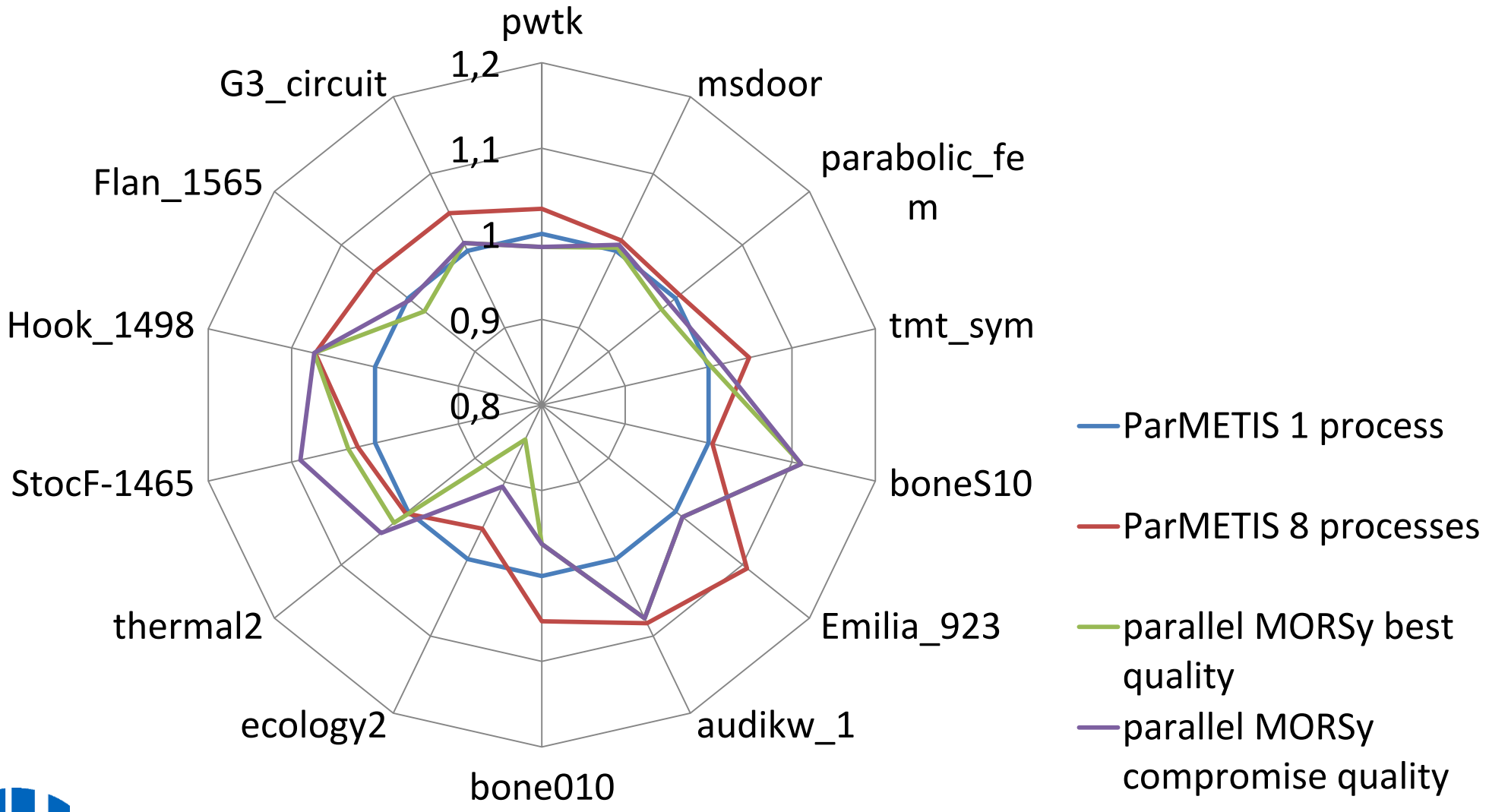
Testing (2)

- Test invironment:

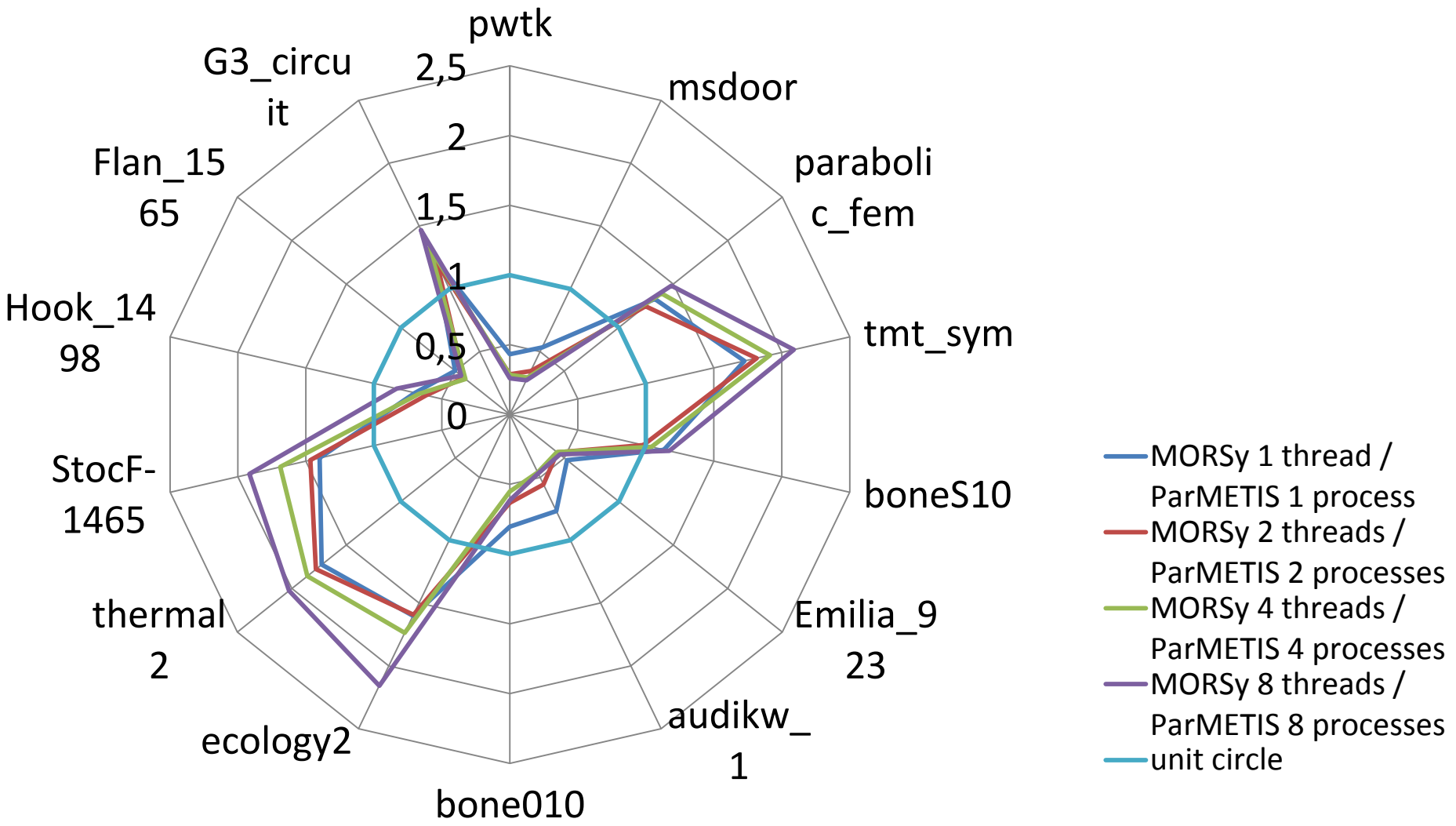
Processor	Intel® Xeon L5630 (2x4-cores, 2.13 GHz)
RAM	24 GB
OS	Windows Server SP2
Compiler	Intel C++ Composer (Intel Parallel Studio XE 2013)

- To compare:
  - Parallel MORSy (OpenMP)
  - ParMETIS 4.0.3 (MPI)

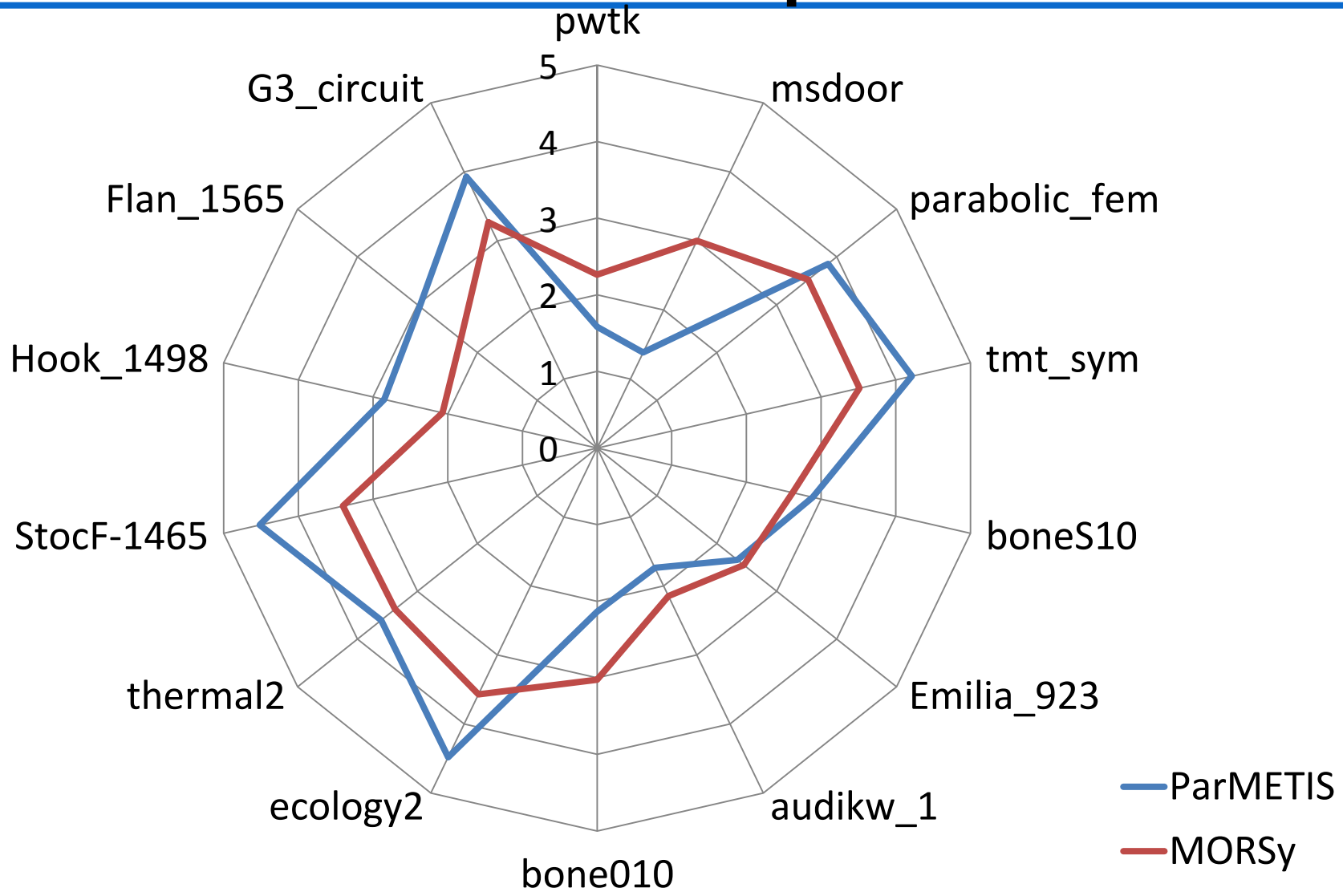
Number of non-zeros in the factor after reordering, relating to ParMETIS



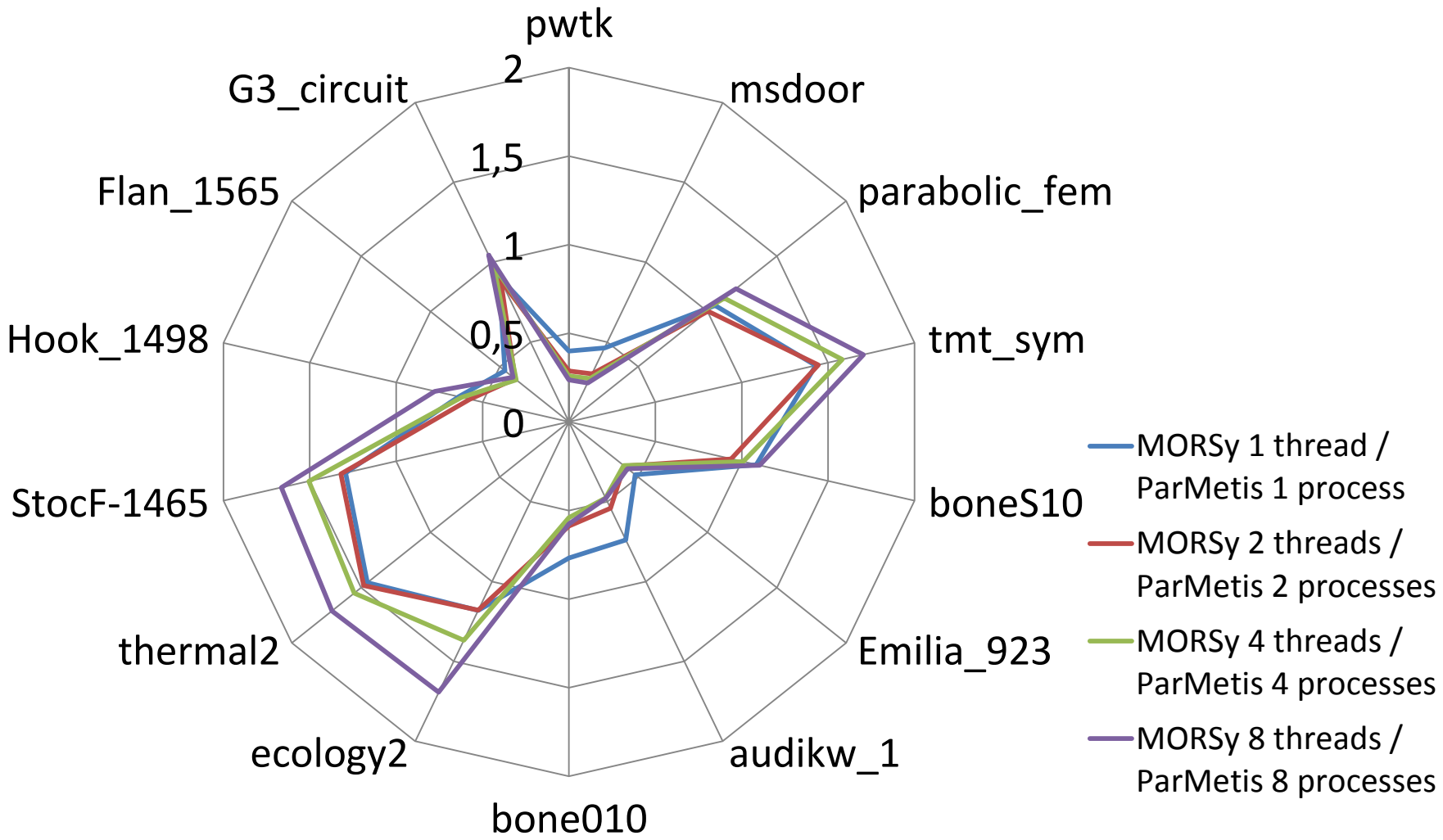
Reordering time, relating to ParMETIS. Task-based parallelism with concurrent queue



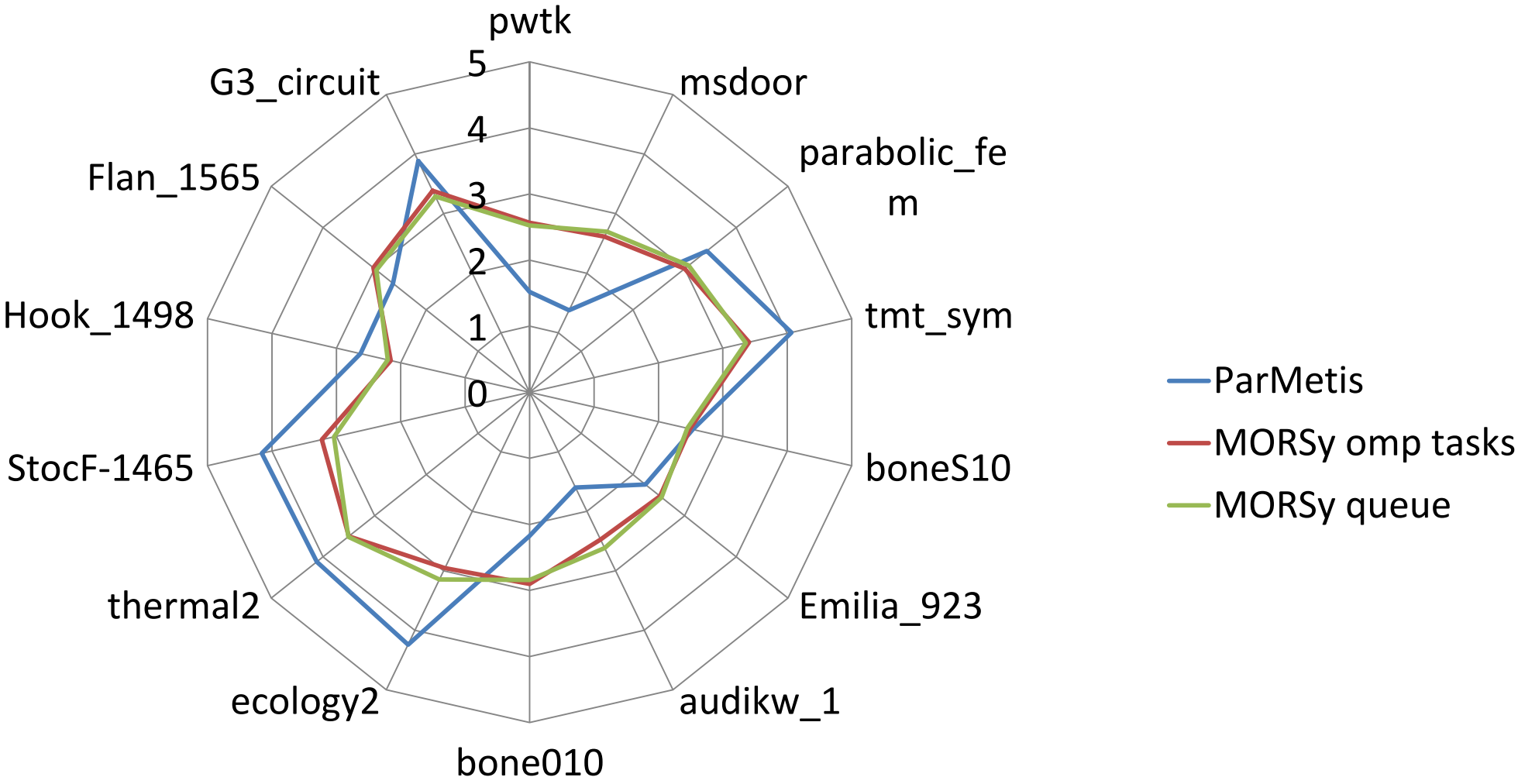
Speedup (8 threads \ processes). Task-based parallelism with concurrent queue



Reordering time, relating to ParMETIS. Task-based parallelism with OpenMP tasks



Speedup (8 threads \ processes). Task-based parallelism with OpenMP tasks



Conclusions

- Experimental results show the competitiveness of our implementation with the widely used ParMETIS library.
- The quality of orderings produced by parallel MORSy is independent of the number of threads. For most test matrices, the number of non-zeros of the resulting factor obtained by parallel MORSy is better than that of ParMETIS.
- In terms of performance, compared to ParMETIS, parallel MORSy works about 2.2 times faster on the half of matrices and about 1.6 times slower on the rest of matrices. MORSy demonstrates suboptimal scalability up to 3.5 times on 8 cores.

Future work

- The main directions of future work:
  - Improving the run time and scalability of our implementation
  - Hybrid MPI + OpenMP/TBB implementation

Q & A

- ?

Thank you for your interest!