AMG Preconditioners based on Parallel Hybrid Coarsening and Bi-objective Matching

PDP 2023

S M Ferdous, PNNL

✉ sm.ferdous@pnnl.gov

Pasqua D’Ambra, IAC,CNR

Fabio Durastante, Uni. of Pisa

Salvatore Filippone, Uni. of Rome-Vergatta

Mahantesh Halappanavar, PNNL

Alex Pothen, Purdue

EoCoE

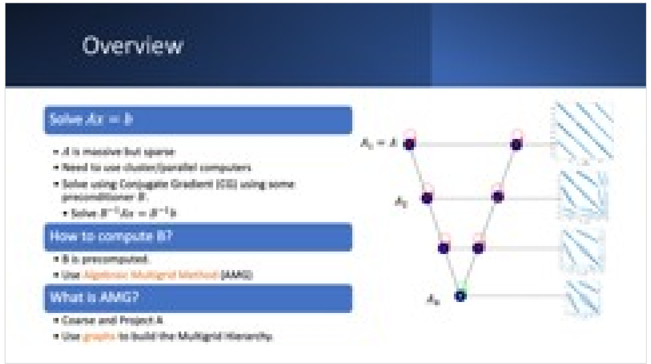
Project ID: 824158

European Commission

Horizon 2020
European Union funding
for Research & Innovation

U.S. DEPARTMENT OF ENERGY

PNNL



Overview

Solve $Ax = b$

- A is massive but sparse
- Need to use cluster/parallel computers
- Solve using Conjugate Gradient (CG) using some preconditioner B.
  - Solve $B^{-1}Ax = B^{-1}b$

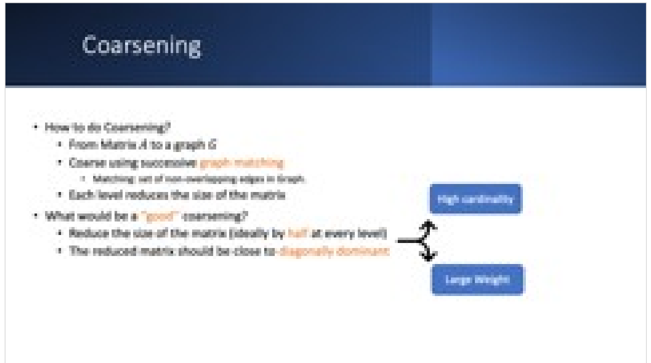
How to compute B?

- B is precomputed.
- Use Algebraic Multigrid Method (AMG)

What is AMG?

- Coarse and Project A
- Use graphs to build the Multigrid Hierarchy

1. Overview



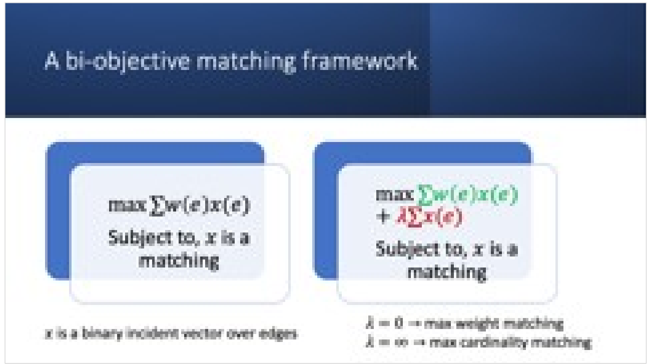
Coarsening

- How to do Coarsening?
  - From Matrix A to a graph G
  - Coarse using successive graph matching
    - Matching: set of non-overlapping edges in Graph.
  - Each level reduces the size of the matrix
- What would be a "good" coarsening?
  - Reduce the size of the matrix (ideally by half at every level)
  - The reduced matrix should be close to diagonally dominant

High cardinality

Large Weight

2. Coarsening



A bi-objective matching framework

$\max \sum w(e)x(e)$

Subject to, x is a matching

x is a binary incident vector over edges

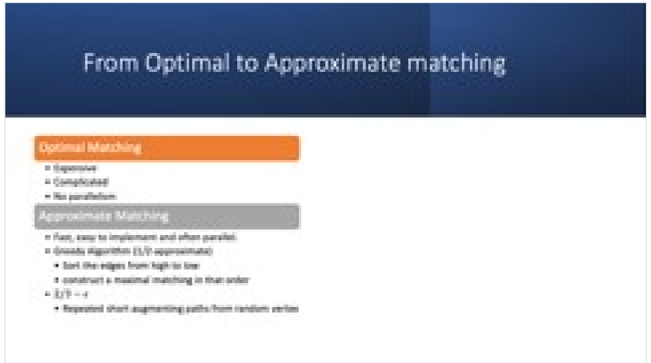
$\max \sum w(e)x(e) + \lambda \sum x(e)$

Subject to, x is a matching

$\lambda = 0 \rightarrow$ max weight matching

$\lambda = \infty \rightarrow$ max cardinality matching

3. Bi-objective matching



From Optimal to Approximate matching

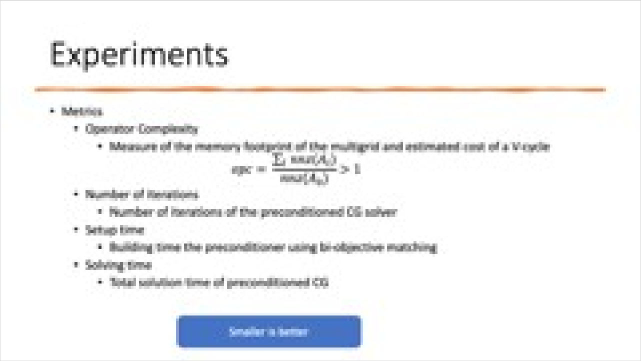
Optimal Matching

- Expensive
- Complicated
- No parallelism

Approximate Matching

- Fast, easy to implement and often parallel
- Greedy Algorithm (1/2-approximate)
  - Sort the edges from high to low
  - construct a maximal matching in that order
- $2/3 - \epsilon$
  - Repeated short augmenting paths from random vertex

4. Approximate Matching



Experiments

- Metrics
  - Operator Complexity
    - Measure of the memory footprint of the multigrid and estimated cost of a V-cycle

$$opc = \frac{\sum_i nnz(A_i)}{nnz(A_0)} > 1$$

  - Number of iterations
    - Number of iterations of the preconditioned CG solver
  - Setup time
    - Building time the preconditioner using bi-objective matching
  - Solving time
    - Total solution time of preconditioned CG

Smaller is better

5. Empirical results

Overview

Solve

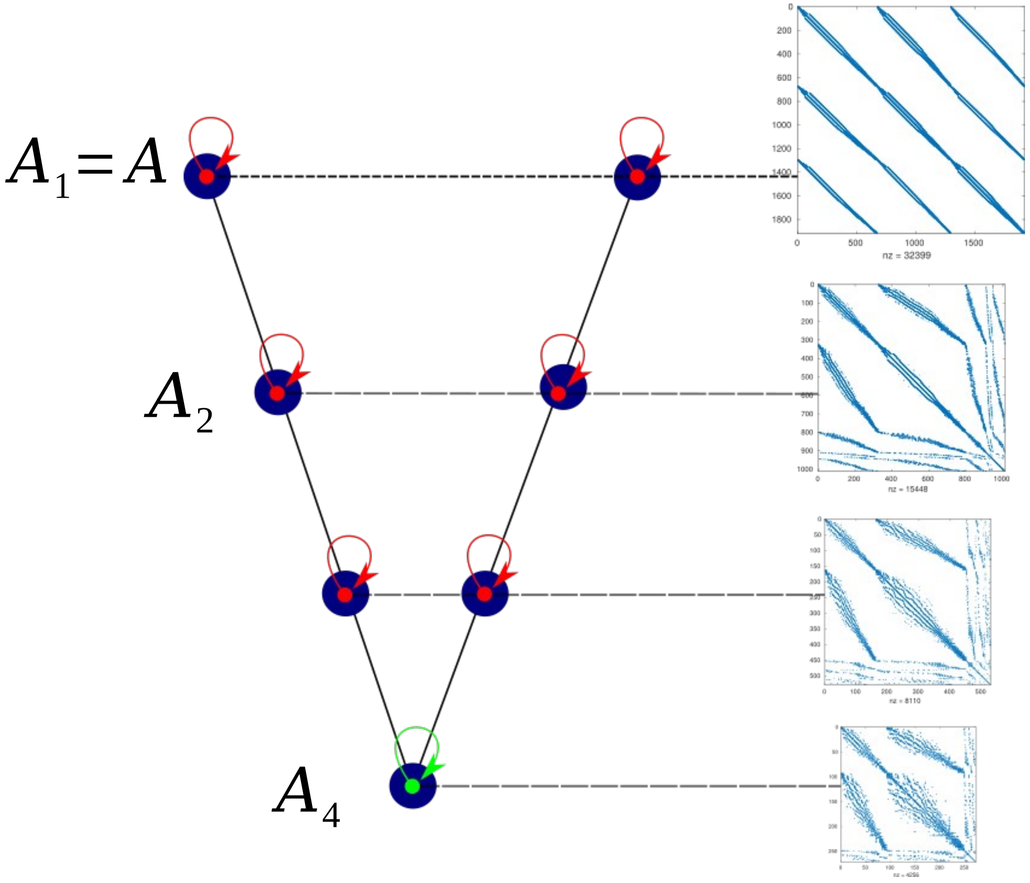
- is massive but sparse
- Need to use cluster/parallel computers
- Solve using Conjugate Gradient (CG) using some preconditioner .
  - Solve

How to compute B?

- B is precomputed.
- Use Algebraic Multigrid Method (AMG)

What is AMG?

- Coarse and Project A
- Use graphs to build the Multigrid Hierarchy.

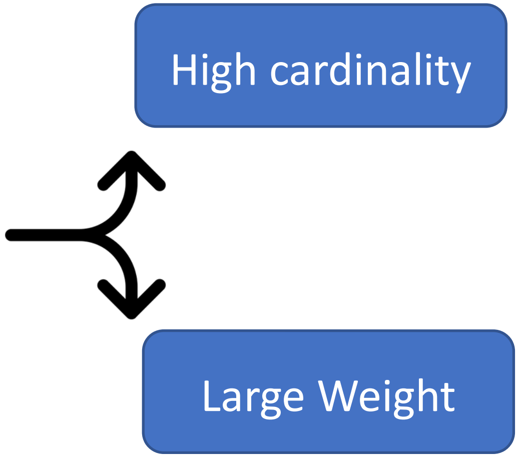


Contributions

- Developed a bi-objective matching framework
- Employed the bi-objective matching to parallel coarsening in AMG
- Experimented on solving anisotropic linear system in multiprocessor.

Coarsening

- How to do Coarsening?
  - From Matrix to a graph
  - Coarse using successive graph matching
    - Matching: set of non-overlapping edges in Graph.
  - Each level reduces the size of the matrix
- What would be a ‘‘good’’ coarsening?
  - Reduce the size of the matrix (ideally by half at every level)
  - The reduced matrix should be close to diagonally dominant



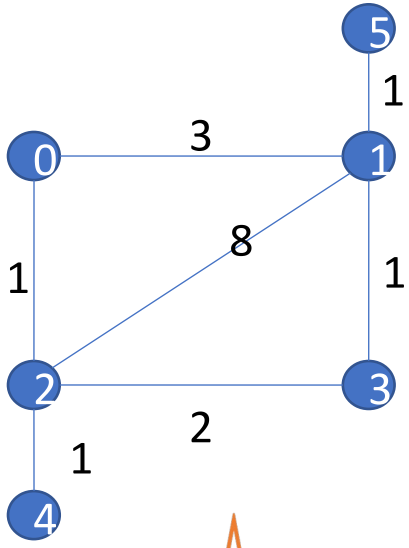
High cardinality

Large Weight

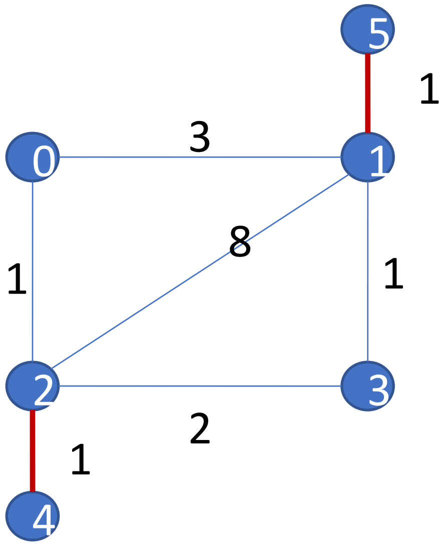
A matching is a set of non-overlapping edges.

$$\begin{pmatrix} 0 & 3 & 1 & 0 & 0 & 0 \\ 3 & 0 & 8 & 0 & 0 & 1 \\ 1 & 8 & 0 & 2 & 1 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

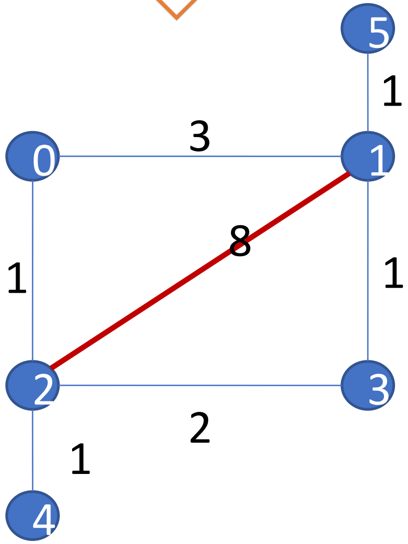
Matrix: A



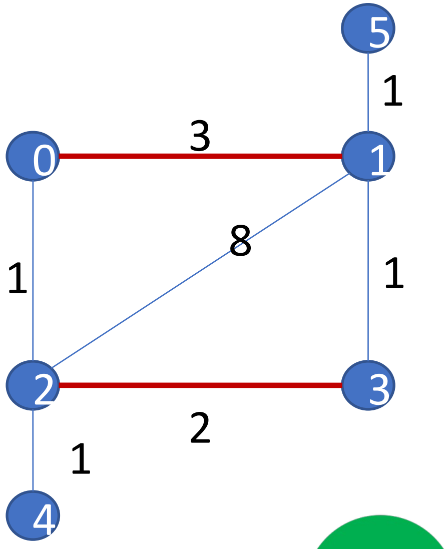
Graph: G



Card: 2
Weight: 2



Card: 1
Weight: 8



Card: 2
Weight: 5 ✓

A bi-objective matching framework

Subject to, is a matching

Subject to, is a matching

is a binary incident vector over edges

max weight matching
max cardinality matching

Properties of-matching

Pareto Optimality

- Both weights and the cardinality are optimal for a parameter > 0.
- Need to solve matching optimally

is parametric to the desired size of matching

- can be a function of max weight of the graph
- set such that matching has certain cardinality guarantee

Subject to, is a matching

$$\lambda = max\left\{\frac{k-1}{2}\gamma - \frac{k+1}{2}\delta , \epsilon\right\}$$

Max weight matching where there is no Augmenting path of length k.
max weight, min weight

From Optimal to Approximate matching

Optimal Matching

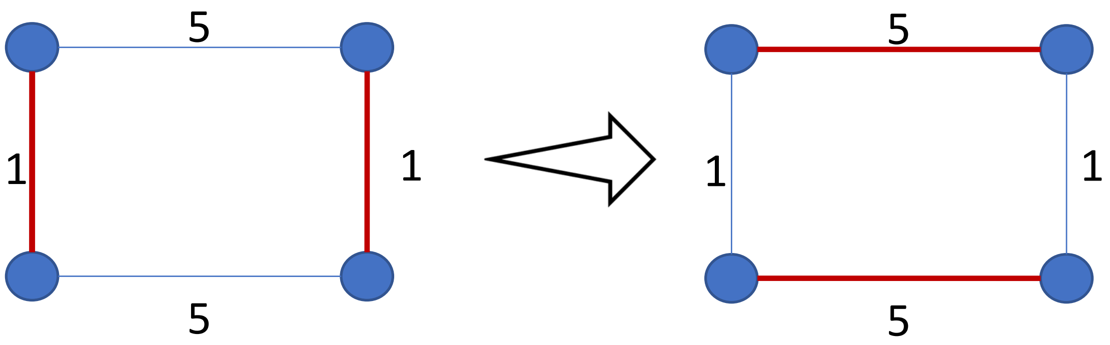
- Expensive
- Complicated
- No parallelism

Approximate Matching

- Fast, easy to implement and often parallel.
- Greedy Algorithm (1/2-approximate)
  - Sort the edges from high to low
  - construct a maximal matching in that order

  - Repeated short augmenting paths from random vertex

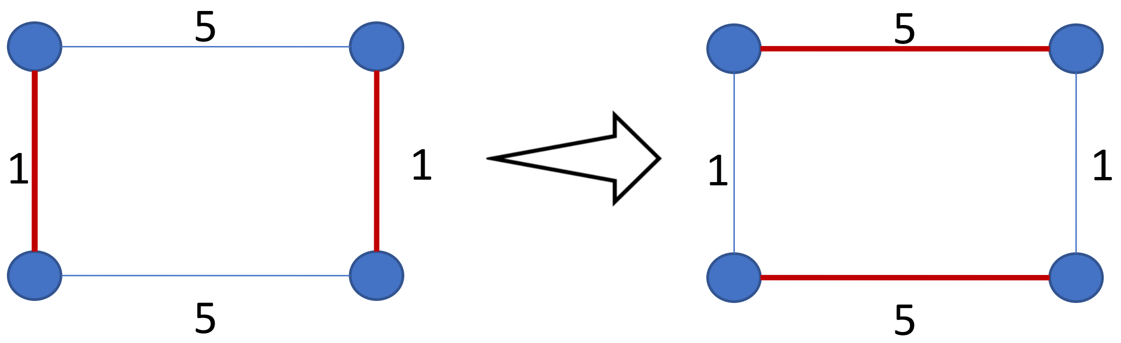
The Randomized 2/3-approximate Algorithm*



A 2-augmentation

*Sanders, Peter, and Seth Pettie. "A simpler linear time 2/3-epsilon approximation for maximum weight matching." (2004).

The Randomized 2/3-approximate Algorithm*



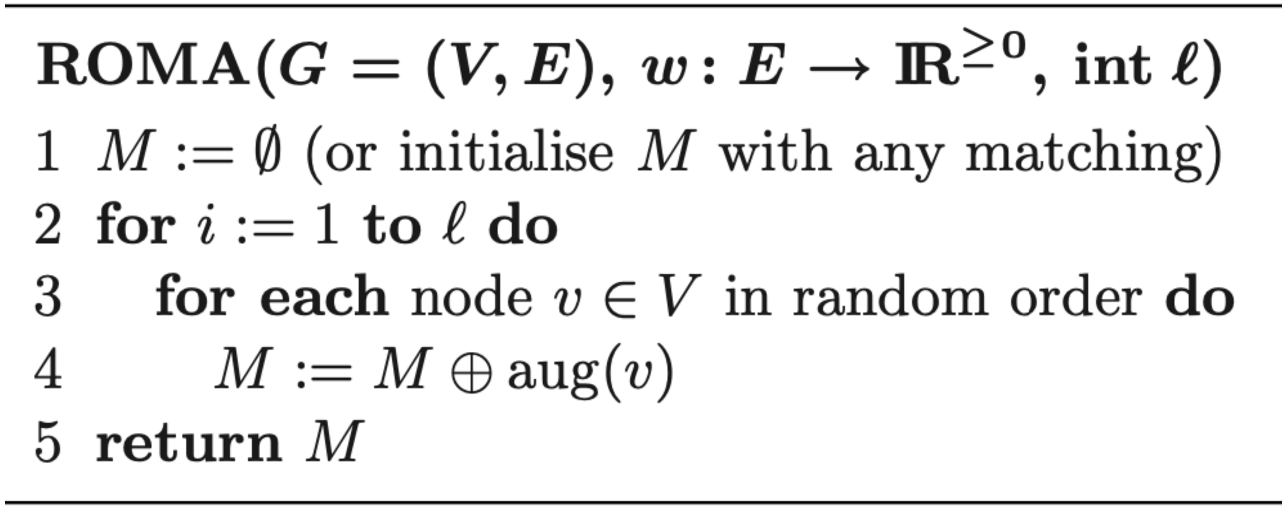
A 2-augmentation

2-augmentation P

- With respect to a matching M
- Weight increasing alternating path
- Number of edges in

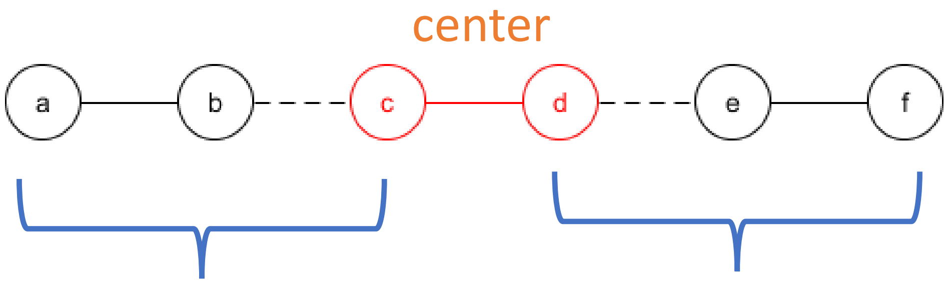
*Sanders, Peter, and Seth Pettie. "A simpler linear time 2/3-epsilon approximation for maximum weight matching." (2004).

The Random Order Augmentation Matching Algorithm ROMA*



```
ROMA($G = (V, E)$, $w : E \to \mathbb{R}^{\geq 0}$, int $\ell$)
$M := \emptyset$ (or initialise $M$ with any matching)
for $i := 1$ to $\ell$ do
    for each node $v \in V$ in random order do
        $M := M \oplus \text{aug}(v)$
return $M$
```

: A 2-augmentation centered at v.



Can also be implemented in a shared memory parallel machine.**

*J. Maue and P. Sanders, “Engineering Algorithms for Approximate Weighted Matching,” in *Experimental Algorithms,2007*
**A. Berge, “A parallel version of the Random Order Augmentation Matching Algorithm,” Master’s thesis, University of Bergen, 2020

Experiments

- Metrics
  - Operator Complexity
    - Measure of the memory footprint of the multigrid and estimated cost of a V-cycle
  - Setup time
    - Build time the preconditioner using bi-objective matching
  - Number of iterations
    - Number of iterations of the preconditioned CG solver
  - Solving time
    - Total solution time of preconditioned CG

↓ lower is better

Problems

- Poisson Benchmark with Axial anisotropy in 2D and 3D
- The boundary value problem

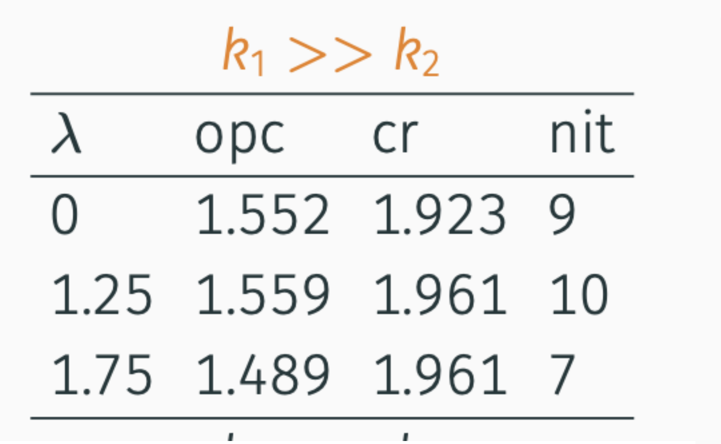
Here,

Machine

- CINECA Marconi 100
  - **Nodes:** 980
  - **Processors:** 2x16 cores IBM POWER9 AC922 at 3.1 GHz
  - **Accelerators:** 4 x NVIDIA Volta V100 GPUs, Nvlink 2.0, 16GB
  - **RAM:** 256 GB/node

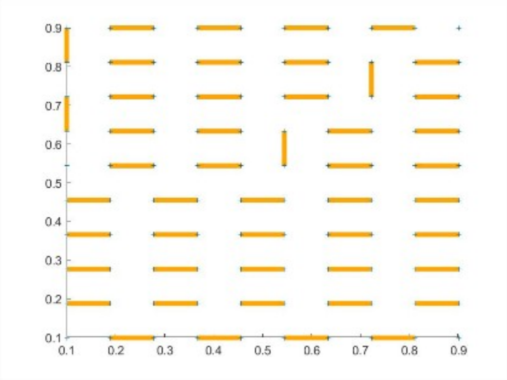
System	Core	Rmax (PFlop/s)
1. Frontiers	8,730,112	1,102.00
2. Fugaku	7,630,848	442.01
3. Lumi	2,220,288	309.10
24. Marconi	347,776	21.64

Effect of

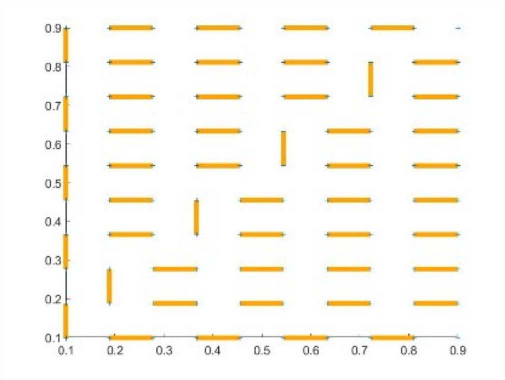


$k_1 >> k_2$

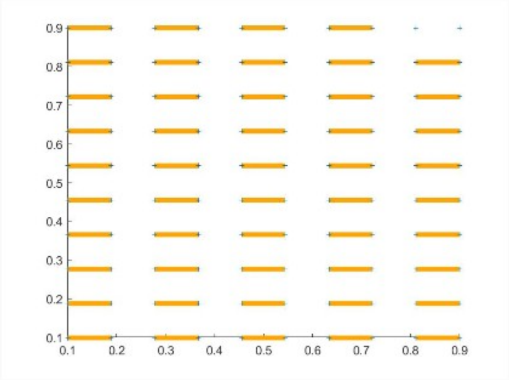
λ	opc	cr	nit
0	1.552	1.923	9
1.25	1.559	1.961	10
1.75	1.489	1.961	7



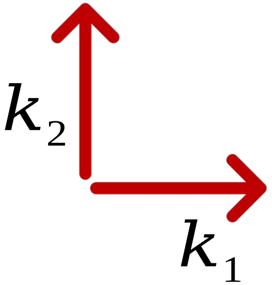
$\lambda = 0,\ k_1 >> k_2$



$\lambda = 1.25,\ k_1 >> k_2$



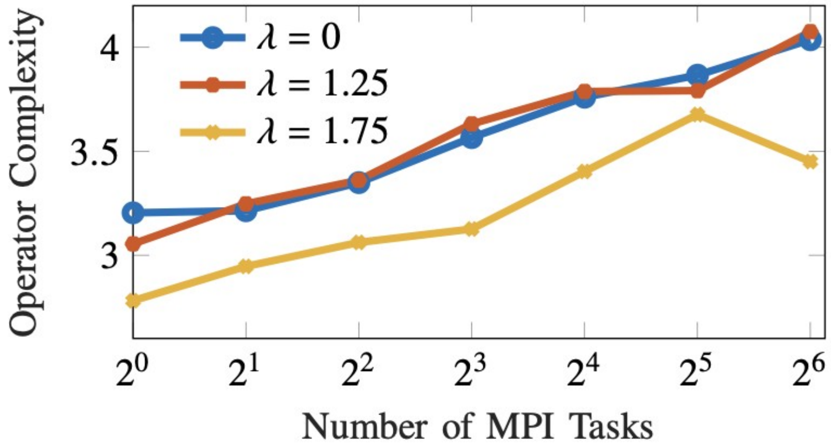
$\lambda = 1.75,\ k_1 >> k_2$



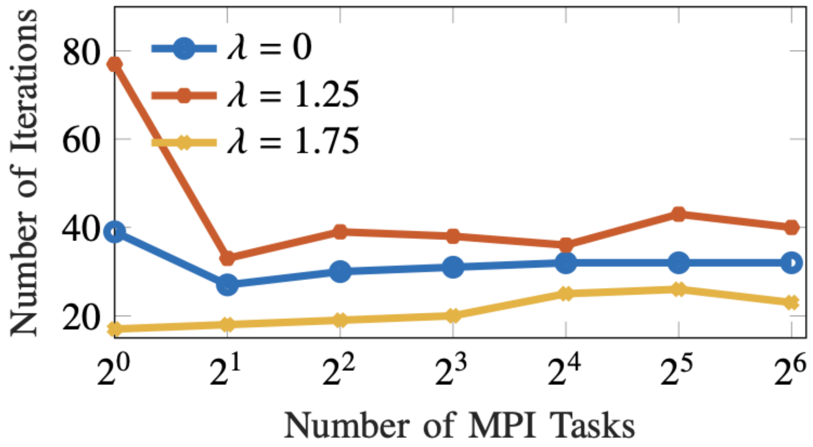
Weak Scaling Analysis

Weak Scaling Setup

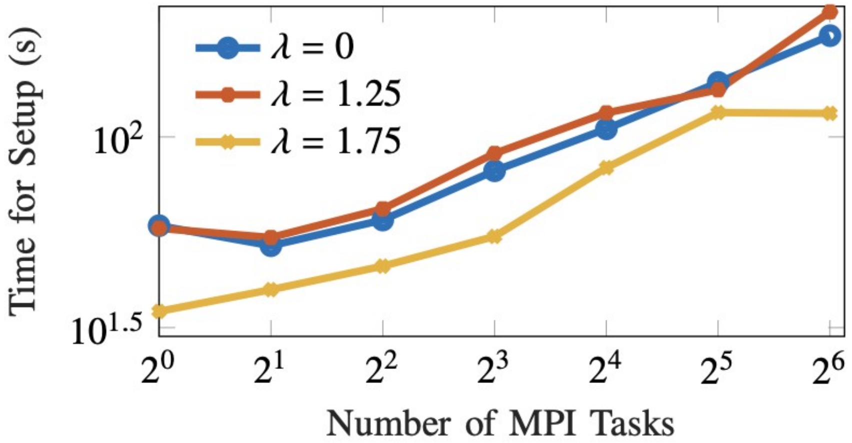
- MPI tasks
- 16 threads per task
- dofs per task
- cores with dofs for the largest problem



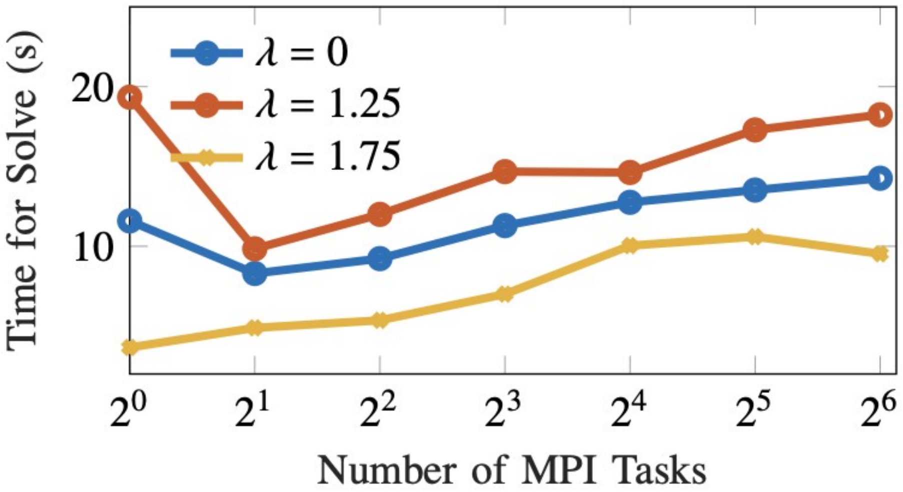
Operator Complexity
4
3.5
3
λ = 0
λ = 1.25
λ = 1.75
2^0
2^1
2^2
2^3
2^4
2^5
2^6
Number of MPI Tasks



Number of Iterations
80
60
40
20
λ = 0
λ = 1.25
λ = 1.75
2^0
2^1
2^2
2^3
2^4
2^5
2^6
Number of MPI Tasks



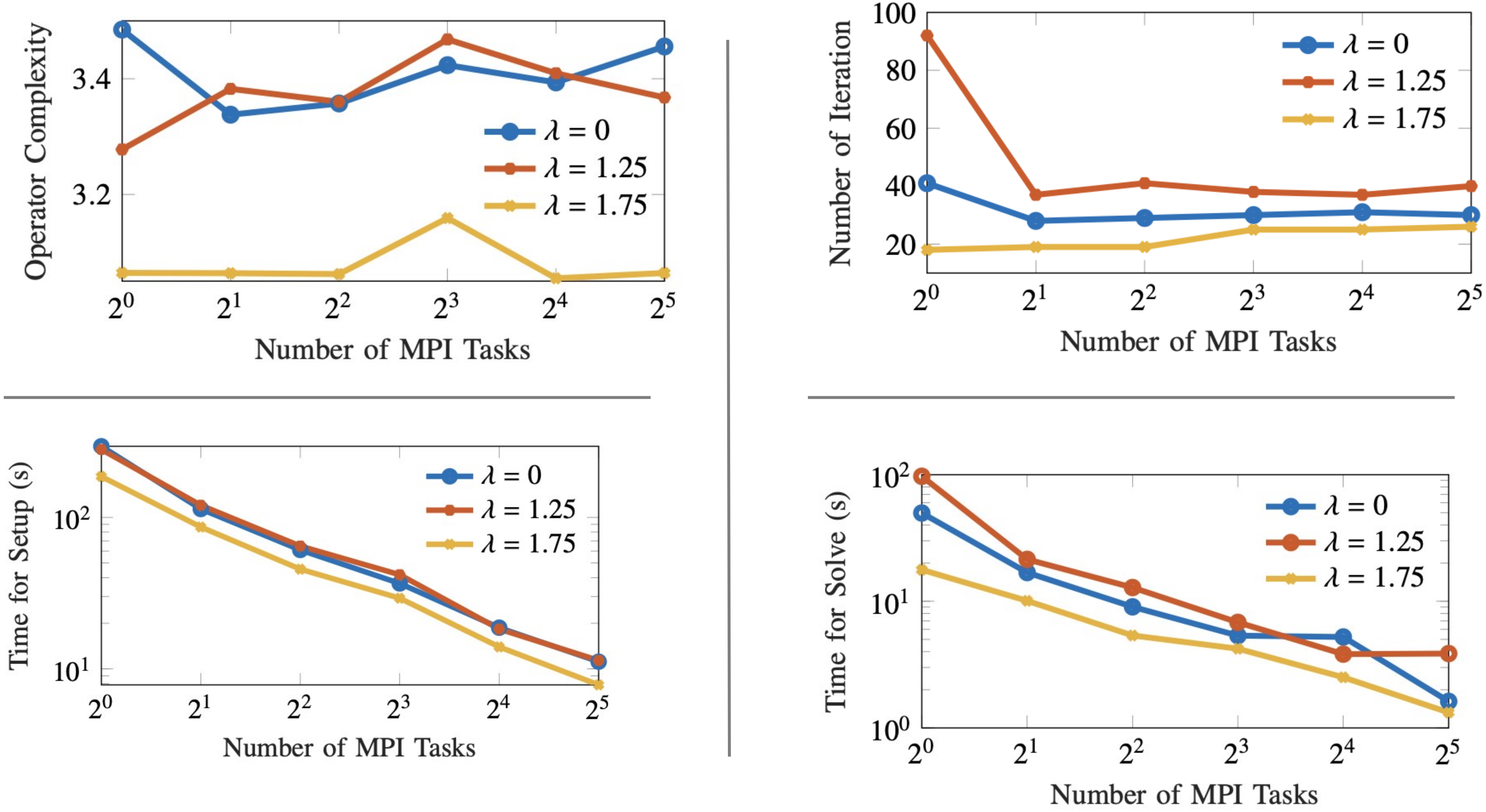
Time for Setup (s)
10^2
$10^{1.5}$
λ = 0
λ = 1.25
λ = 1.75
2^0
2^1
2^2
2^3
2^4
2^5
2^6
Number of MPI Tasks



Time for Solve (s)
20
10
λ = 0
λ = 1.25
λ = 1.75
2^0
2^1
2^2
2^3
2^4
2^5
2^6
Number of MPI Tasks

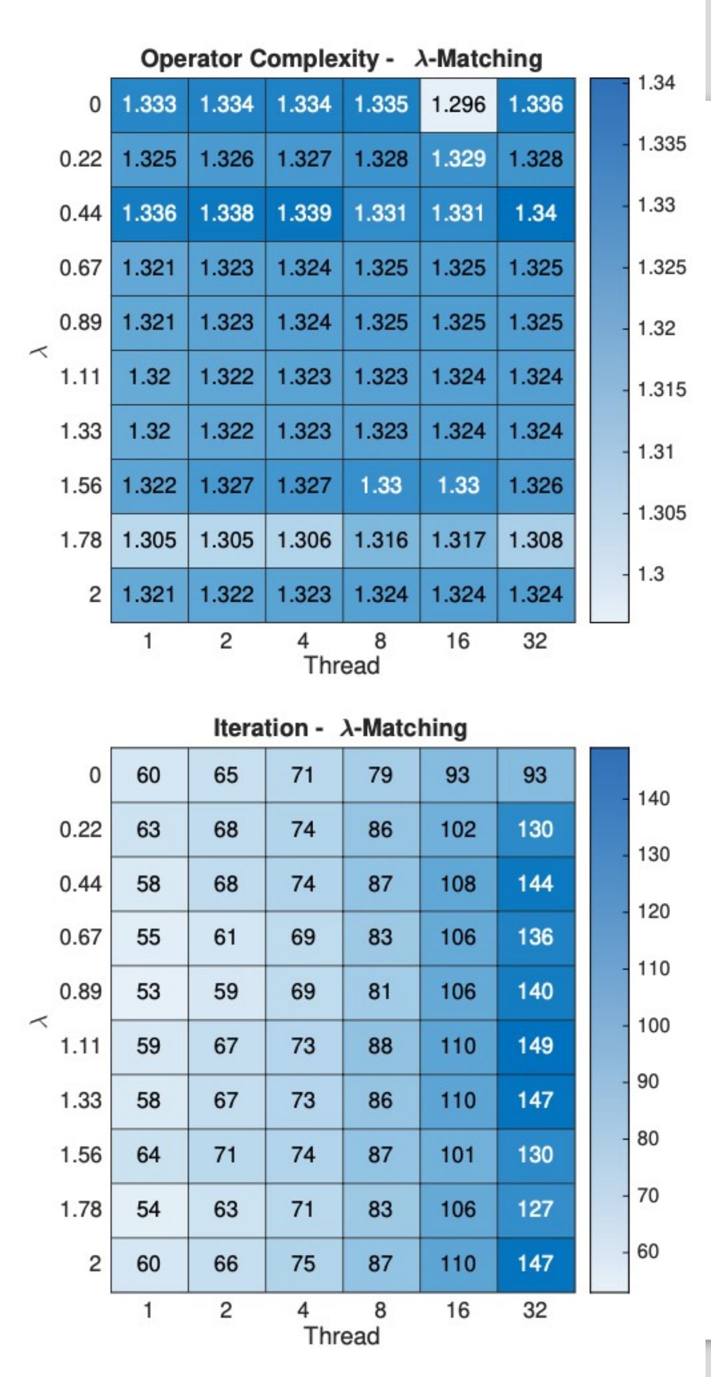
Strong Scaling Analysis

- Setup
  - Number of dofs
  - MPI processes
  - Each task with 16 threads


Operator Complexity
3.4
3.2
2^0
2^1
2^2
2^3
2^4
2^5
Number of MPI Tasks
$\lambda = 0$
$\lambda = 1.25$
$\lambda = 1.75$
Number of Iteration
100
80
60
40
20
2^0
2^1
2^2
2^3
2^4
2^5
Number of MPI Tasks
$\lambda = 0$
$\lambda = 1.25$
$\lambda = 1.75$
Time for Setup (s)
10^2
10^1
2^0
2^1
2^2
2^3
2^4
2^5
Number of MPI Tasks
$\lambda = 0$
$\lambda = 1.25$
$\lambda = 1.75$
Time for Solve (s)
10^2
10^1
10^0
2^0
2^1
2^2
2^3
2^4
2^5
Number of MPI Tasks
$\lambda = 0$
$\lambda = 1.25$
$\lambda = 1.75$

Comparison with other algorithms

- Benchmark Algorithms
  - Bi-objective Matching (our)
  - Parmatch (greedy matching as aggregator)
  - VBM
  - Falgout
  - HMIS1
  - ML

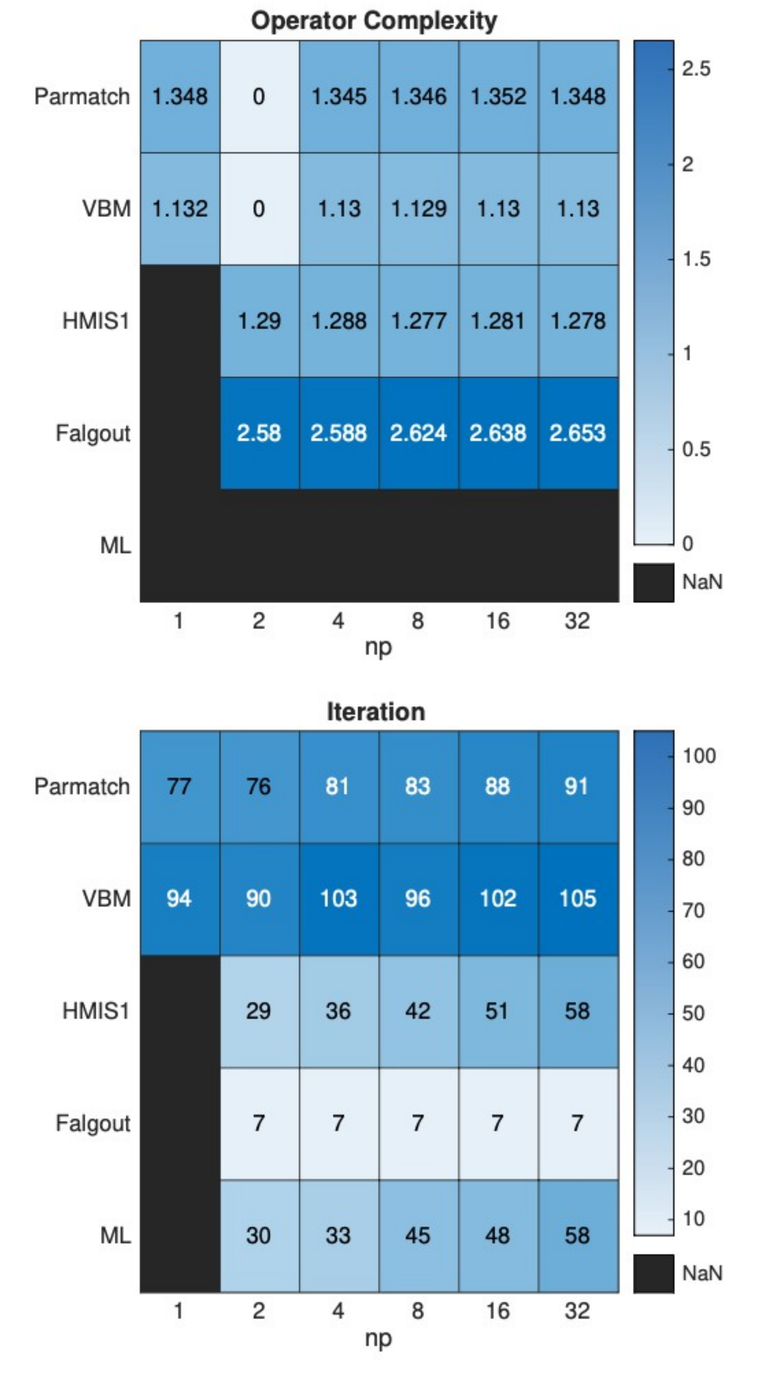


Operator Complexity - λ-Matching

λ \ Thread	1	2	4	8	16	32
0	1.333	1.334	1.334	1.335	1.296	1.336
0.22	1.325	1.326	1.327	1.328	1.329	1.328
0.44	1.336	1.338	1.339	1.331	1.331	1.34
0.67	1.321	1.323	1.324	1.325	1.325	1.325
0.89	1.321	1.323	1.324	1.325	1.325	1.325
1.11	1.32	1.322	1.323	1.323	1.324	1.324
1.33	1.32	1.322	1.323	1.323	1.324	1.324
1.56	1.322	1.327	1.327	1.33	1.33	1.326
1.78	1.305	1.305	1.306	1.316	1.317	1.308
2	1.321	1.322	1.323	1.324	1.324	1.324

Iteration - λ-Matching

λ \ Thread	1	2	4	8	16	32
0	60	65	71	79	93	93
0.22	63	68	74	86	102	130
0.44	58	68	74	87	108	144
0.67	55	61	69	83	106	136
0.89	53	59	69	81	106	140
1.11	59	67	73	88	110	149
1.33	58	67	73	86	110	147
1.56	64	71	74	87	101	130
1.78	54	63	71	83	106	127
2	60	66	75	87	110	147

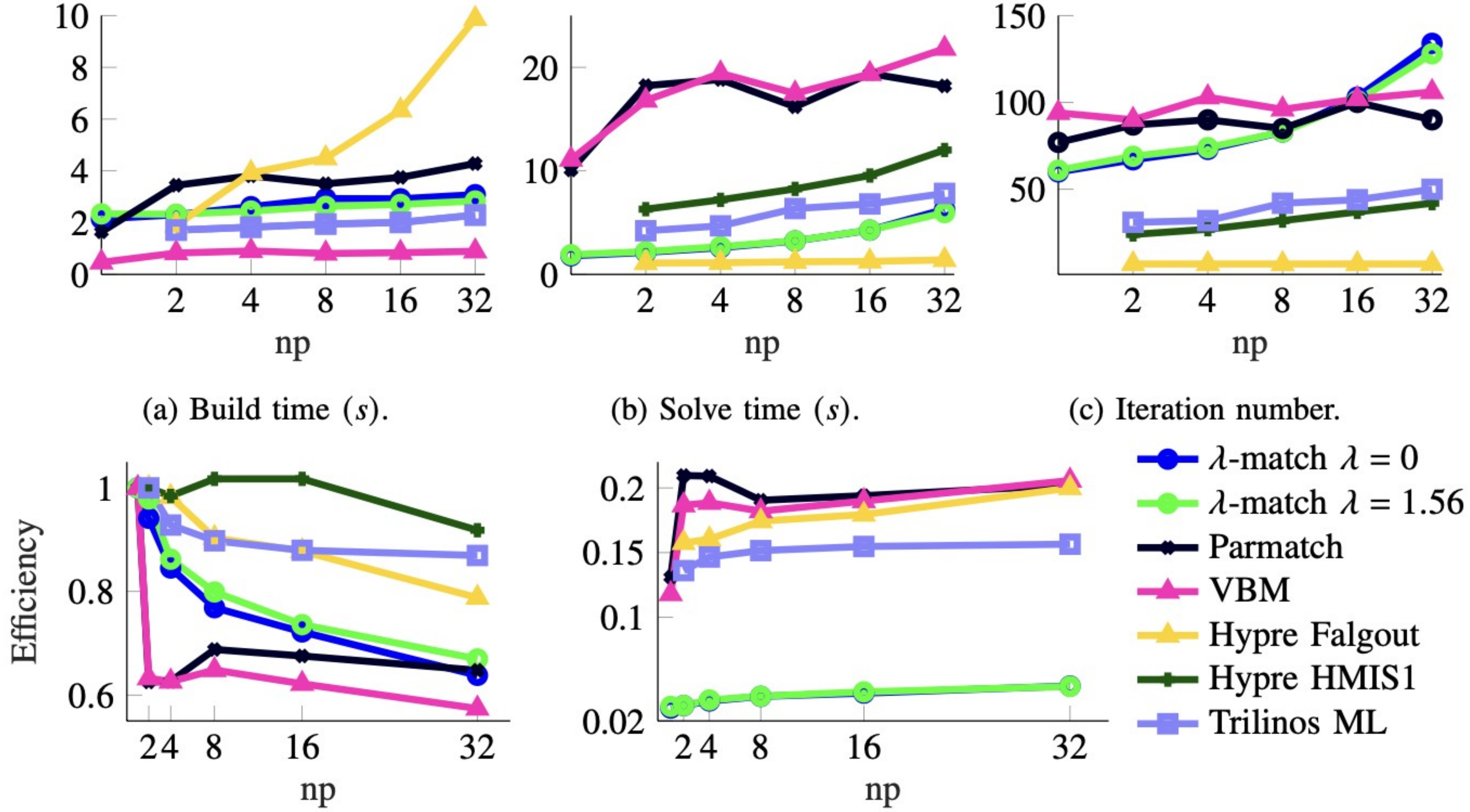


Operator Complexity

np	1	2	4	8	16	32
Parmatch	1.348	0	1.345	1.346	1.352	1.348
VBM	1.132	0	1.13	1.129	1.13	1.13
HMIS1	NaN	1.29	1.288	1.277	1.281	1.278
Falgout	NaN	2.58	2.588	2.624	2.638	2.653
ML	NaN	NaN	NaN	NaN	NaN	NaN

Iteration

np	1	2	4	8	16	32
Parmatch	77	76	81	83	88	91
VBM	94	90	103	96	102	105
HMIS1	NaN	29	36	42	51	58
Falgout	NaN	7	7	7	7	7
ML	NaN	30	33	45	48	58



(a) Build time (s).

(b) Solve time (s).

(c) Iteration number.

(d) Time per iteration (Efficiency)

(e) Time per iteration (s).

Thank You!

💻 https://smferdous1.github.io/

✉ sm.ferdous@pnnl.gov