Evolving Algebraic Multigrid Methods Using Grammar-Guided Genetic Programming

Dinesh Parthasarathy[1], Wayne Bradford Mitchell[2], Harald Köstler[1]

1. Friedrich-Alexander-Universität Erlangen-Nürnberg

2. Lawrence Livermore National Laboratory

Background

- Designing an efficient multigrid method is non-trivial.

- The selection of algorithmic components plays a crucial role in determining its efficiency.

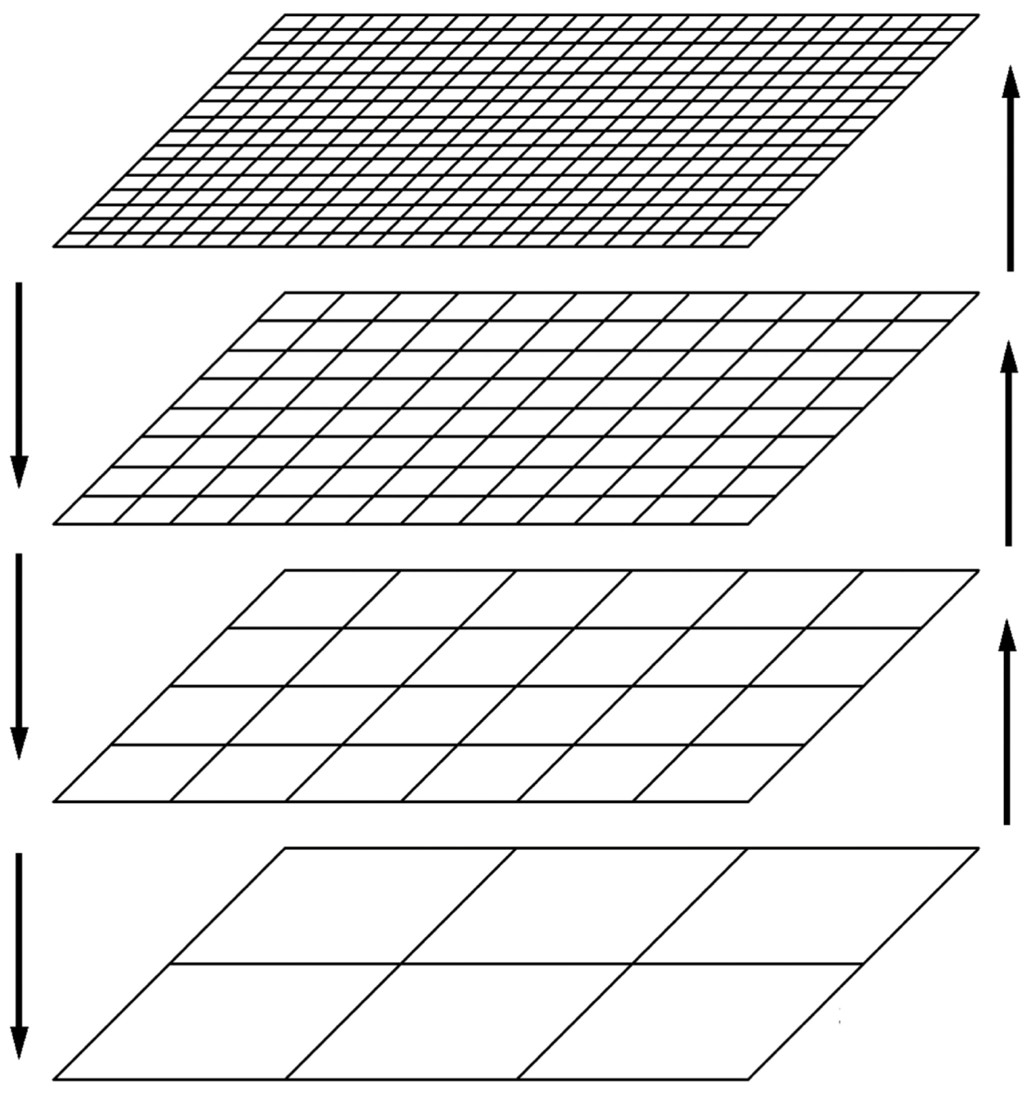
- Many existing efforts leverage AI to optimise individual components, such as:
  - Learning optimal multigrid smoothers via neural networks (Huang et al., 2023).
  - Learning optimal prolongation operators using GNN (Luz et al., ICML 2020).
  - Optimising coarsening schemes using reinforcement learning (Taghibakhshi et al., 2021).
  - Learning optimal relaxation schemes and weights (Nytko., CMCIM 2024).

Complementary approach
Construct efficient multigrid cycles from a set of available multigrid components.

Overview

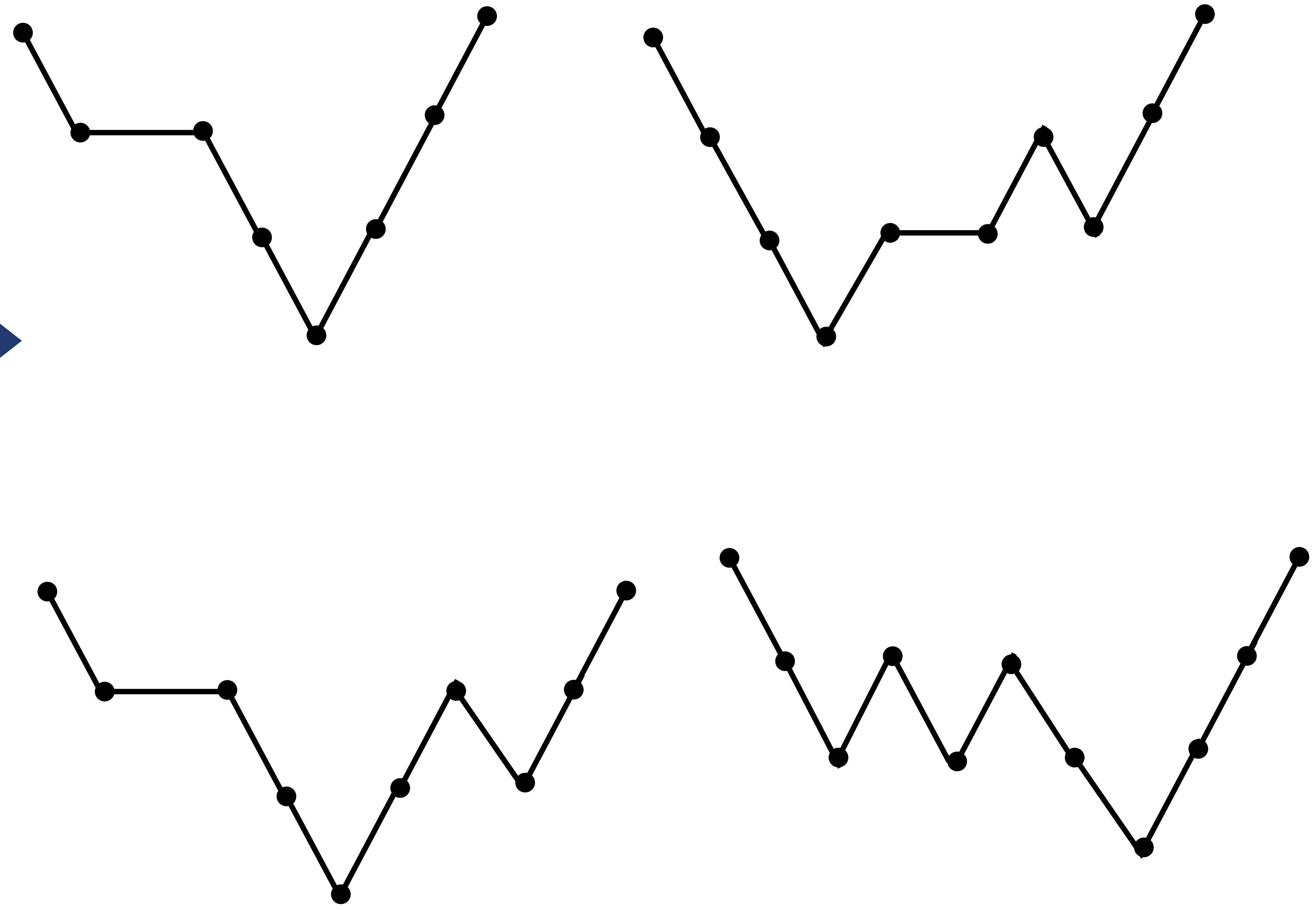
$$\epsilon u_{xx} + u_{yy} + u_{zz} = f \ \textbf{ in } \ \Omega$$
$$u = 0 \ \textbf{ on } \ \partial\Omega$$

+



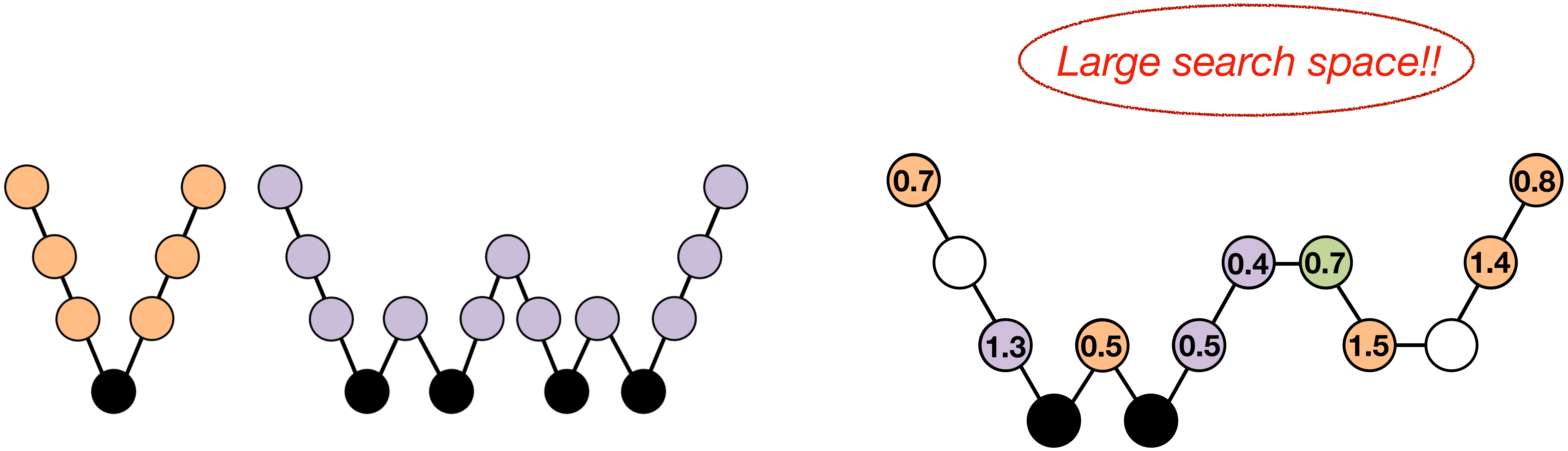
Gauss-Seidel Fwd.
Gauss-Seidel Bwd.
Gauss-Seidel Symm.
Jacobi
Chebyshev

0.1
0.2
0.3
..
..
1.8
1.9



Dinesh Parthasarathy
dinesh.parthasarathy@fau.de

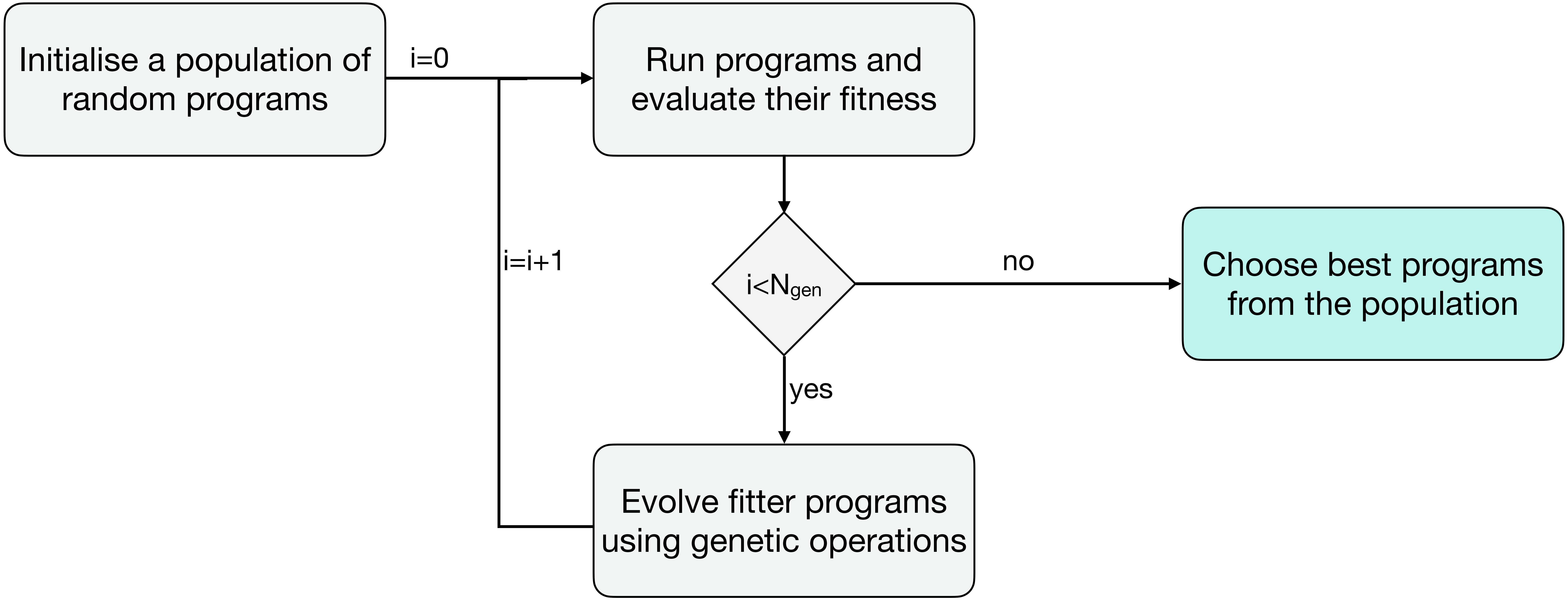
Motivation



Standard Cycles

Flexible Cycles

Genetic Programming – Primer



Genetic Programming – Grammar Guided Approach

$$
\begin{array}{rcl}
\langle S \rangle & \models & \langle s_h \rangle \\
\langle s_h \rangle & \models & \textsc{iterate}(\langle c_h \rangle,\ \omega,\ \langle \mathcal{P} \rangle)\ \mid\ (u_h^0,\ f_h,\ \lambda,\ \lambda) \\
\langle s_h \rangle & \models & \textsc{iterate}(\textsc{apply}(\langle B_h \rangle,\ \langle c_h \rangle),\ \omega,\ \langle \mathcal{P} \rangle) \\
\langle s_h \rangle & \models & \textsc{iterate}(\textsc{coarse-grid-correction}(I_{2h}^{h},\ \langle s_{2h} \rangle),\ \omega,\ \langle \mathcal{P} \rangle) \\
\langle c_h \rangle & \models & \textsc{residual}(A_h,\ \langle s_h \rangle) \\
\langle B_h \rangle & \models & \textsc{inverse}(A_h^{+}) \text{ with } A_h = A_h^{+} + A_h^{-} \\
\langle c_{2h} \rangle & \models & \textsc{residual}(A_{2h},\ \langle s_{2h} \rangle) \\
\langle c_{2h} \rangle & \models & \textsc{coarse-cycle}(A_{2h},\ u_{2h}^{0},\ \textsc{apply}(I_h^{2h},\ \langle c_h \rangle)) \\
\langle s_{2h} \rangle & \models & \textsc{iterate}(\langle c_{2h} \rangle,\ \omega,\ \langle \mathcal{P} \rangle) \\
\langle s_{2h} \rangle & \models & \textsc{iterate}(\textsc{apply}(\langle B_{2h} \rangle,\ \langle c_{2h} \rangle),\ \omega,\ \langle \mathcal{P} \rangle) \\
\langle s_{2h} \rangle & \models & \textsc{iterate}(\textsc{apply}(I_{4h}^{2h},\ \langle c_{4h} \rangle),\ \omega,\ \lambda) \\
\langle B_{2h} \rangle & \models & \textsc{inverse}(A_{2h}^{+}) \text{ with } A_{2h} = A_{2h}^{+} + A_{2h}^{-} \\
\langle c_{4h} \rangle & \models & \textsc{apply}(A_{4h}^{-1},\ \textsc{apply}(I_{2h}^{4h},\ \langle c_{2h} \rangle)) \\
\langle \mathcal{P} \rangle & \models & \textsc{partitioning}\ \mid\ \lambda
\end{array}
$$

(Schmitt et al., 2021)

Start with $\langle S \rangle$, and replace expressions until no $\langle\ .\ \rangle$ remains

$$\langle s_h \rangle$$

$$\downarrow$$

ITERATE(APPLY($D_h^{-1}, \langle c_h \rangle$), ω, λ)

$$\downarrow$$

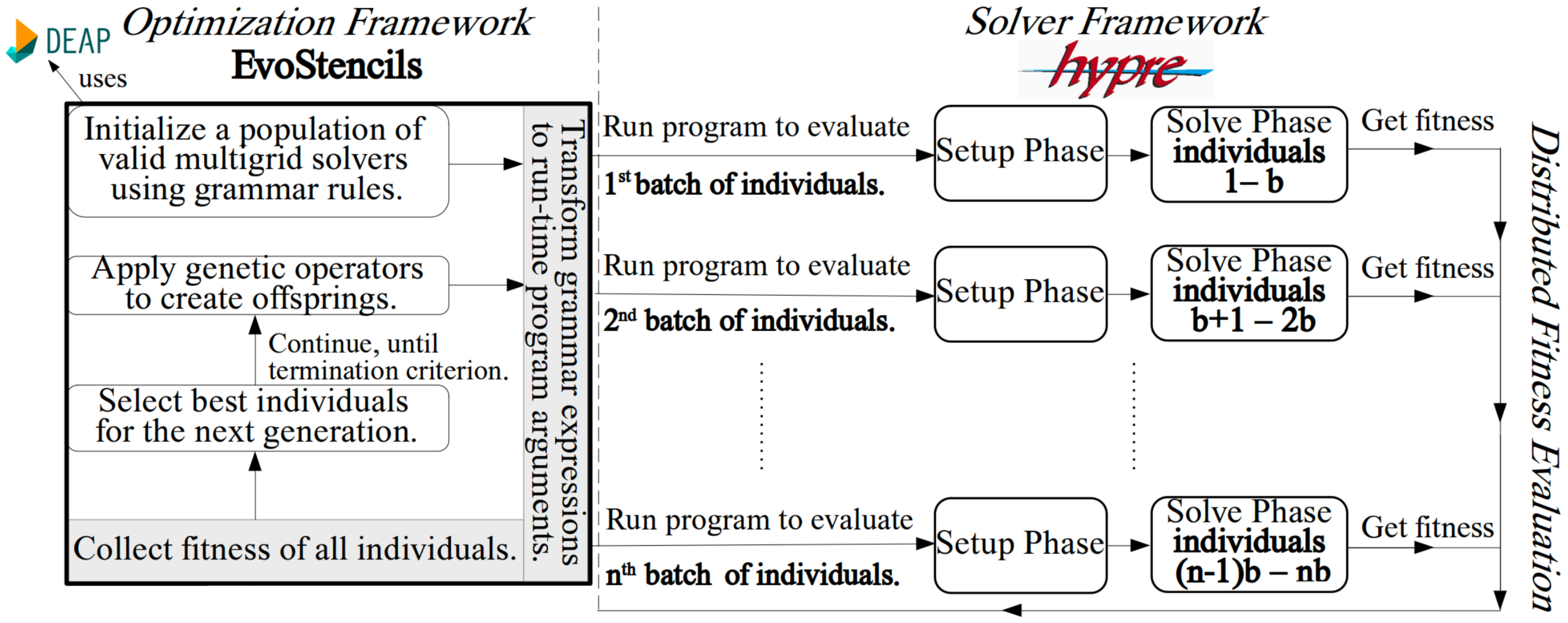
ITERATE(APPLY(D_h^{-1}, RESIDUAL($A_h, \langle s_h \rangle$)), ω, λ)

$$\downarrow$$

ITERATE(APPLY(D_h^{-1}, RESIDUAL($A_h, (u_h^0, f_h, \lambda, \lambda)$)), ω, λ)

1 iteration of ω-Jacobi

Software


DEAP
Optimization Framework
EvoStencils
uses
Initialize a population of valid multigrid solvers using grammar rules.
Apply genetic operators to create offsprings.
Continue, until termination criterion.
Select best individuals for the next generation.
Collect fitness of all individuals.
Transform grammar expressions to run-time program arguments.
Solver Framework
hypre
Run program to evaluate
1st batch of individuals.
Setup Phase
Solve Phase individuals 1 – b
Get fitness
Run program to evaluate
2nd batch of individuals.
Setup Phase
Solve Phase individuals b+1 – 2b
Get fitness
Run program to evaluate
nth batch of individuals.
Setup Phase
Solve Phase individuals (n-1)b – nb
Get fitness
Distributed Fitness Evaluation

Experimental Setup

Dinesh Parthasarathy
dinesh.parthasarathy@fau.de

Problem

- $\epsilon u_{xx} + u_{yy} + u_{zz} = f$ **in** Ω

  $u = 0$ **on** $\partial\Omega$

- The system of equations is built using a standard 7-point stencil in a unit cube.

- Stopping tolerance

  Relative residual norm $= 10^{-8}$

Dinesh Parthasarathy
dinesh.parthasarathy@fau.de

Optimization settings

Smoothers	Gauss-Seidel forward, Gauss-Seidel backward, Jacobi
Relaxation weights	(0.1, 0.15, 0.2, ..., 1.9)
Scaling factors	(0.1, 0.15, 0.2, ..., 1.9)
Coarsening strategy	HMIS algorithm
Interpolation	Extended+i
Restriction	Interpolation transpose
Coarse grid solver	Gaussian elimination

AMG components

Evolutionary algorithm	$(\mu + \lambda)$
Generations	100
μ*(population)*	256
λ*(offsprings)*	256
Initial pop.	2048
MPI processes	64
Sorting alg.	NSGA-II

GP parameters

ϵ	0.001
f	0
N_d	100
$u^{(0)}$	$rand$

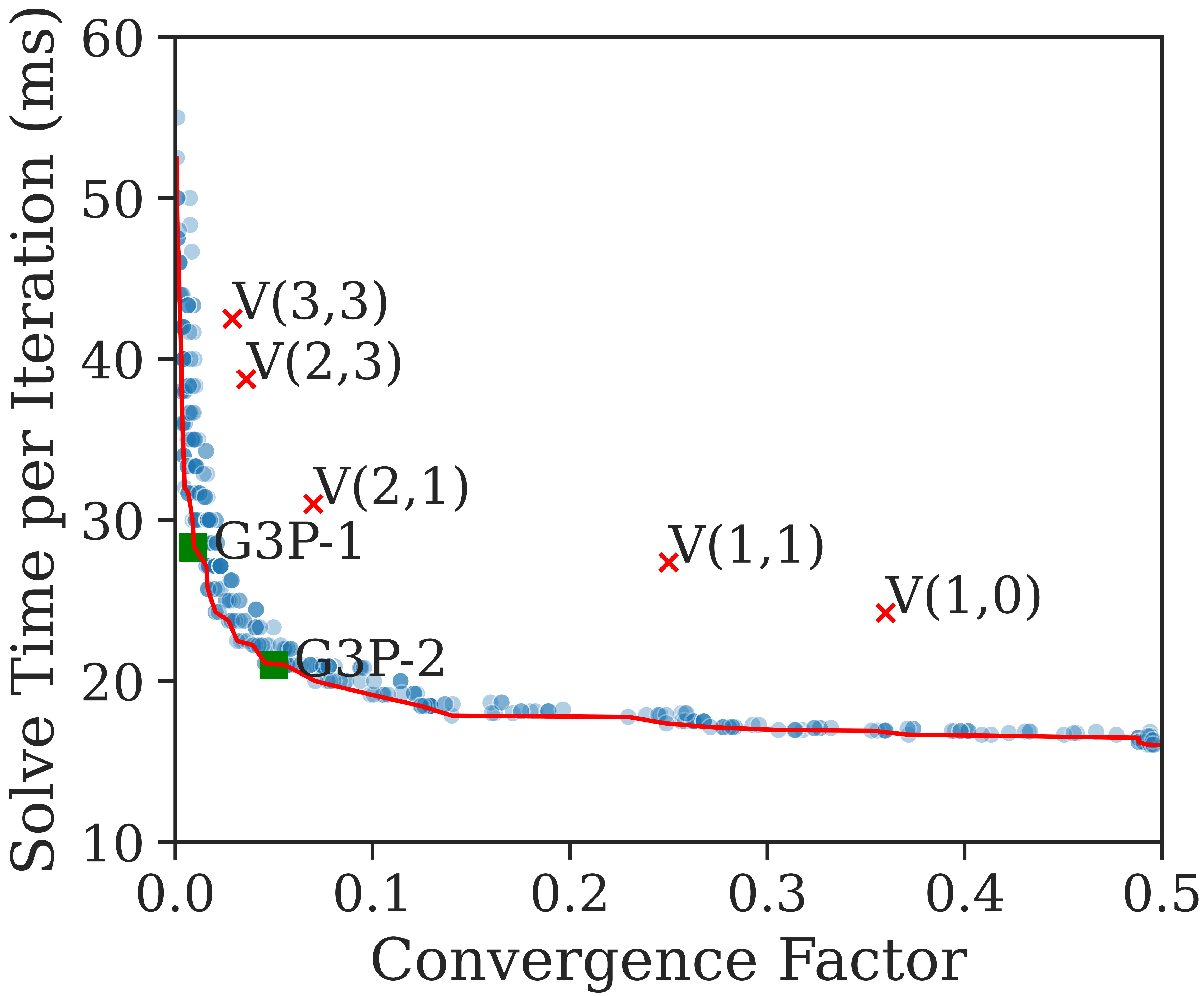
Problem parameters

Results

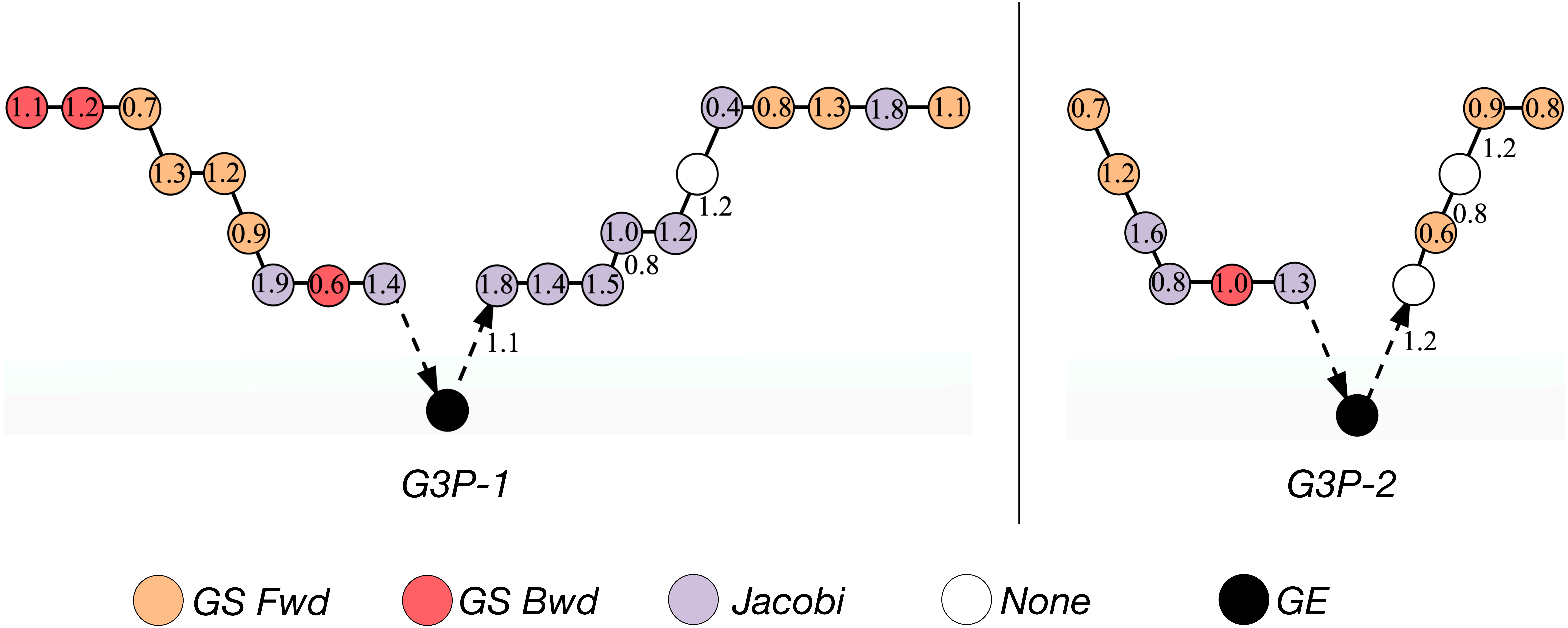
Dinesh Parthasarathy
dinesh.parthasarathy@fau.de

Pareto distribution

W(1,1)



Solver structure



Dinesh Parthasarathy
dinesh.parthasarathy@fau.de

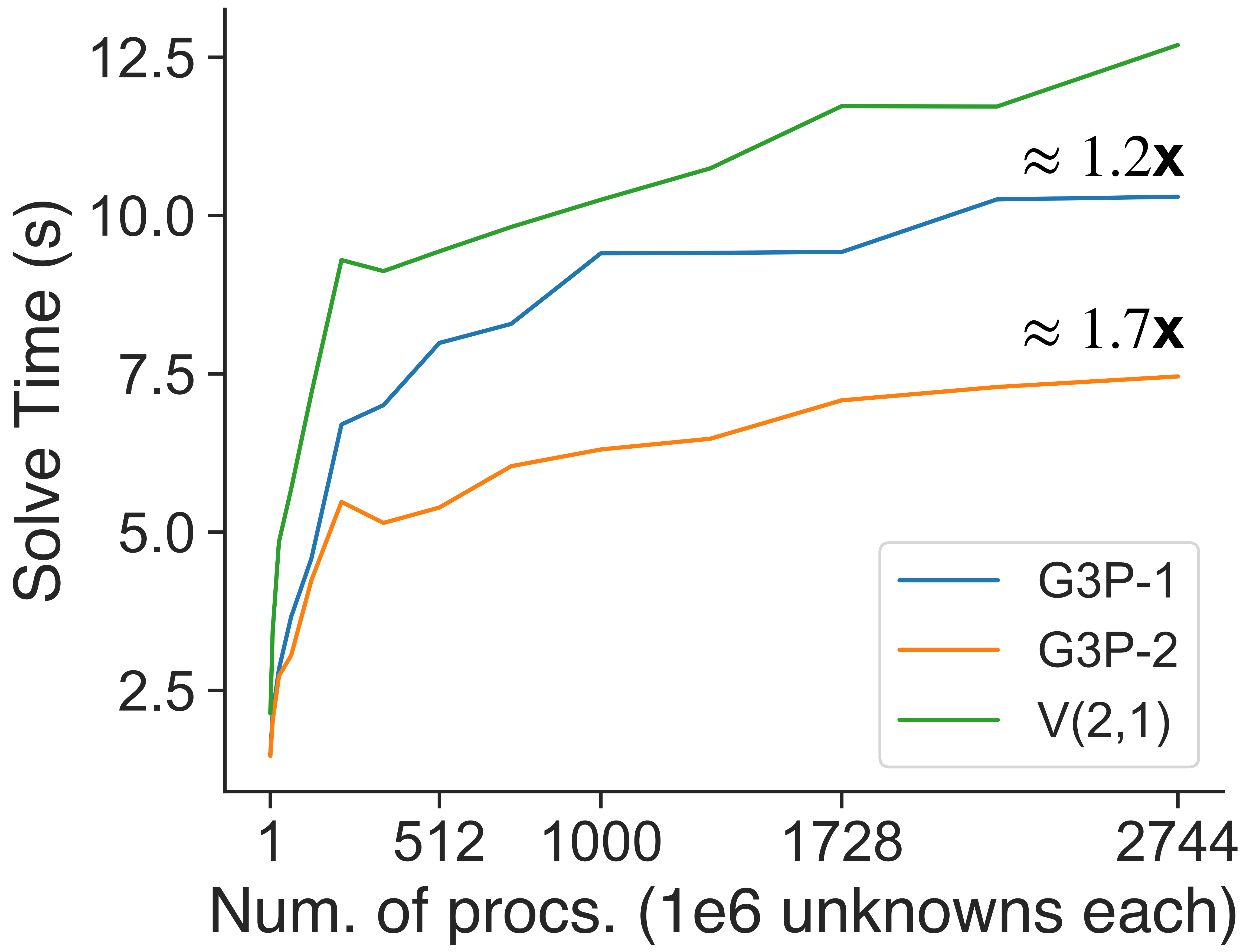
Performance

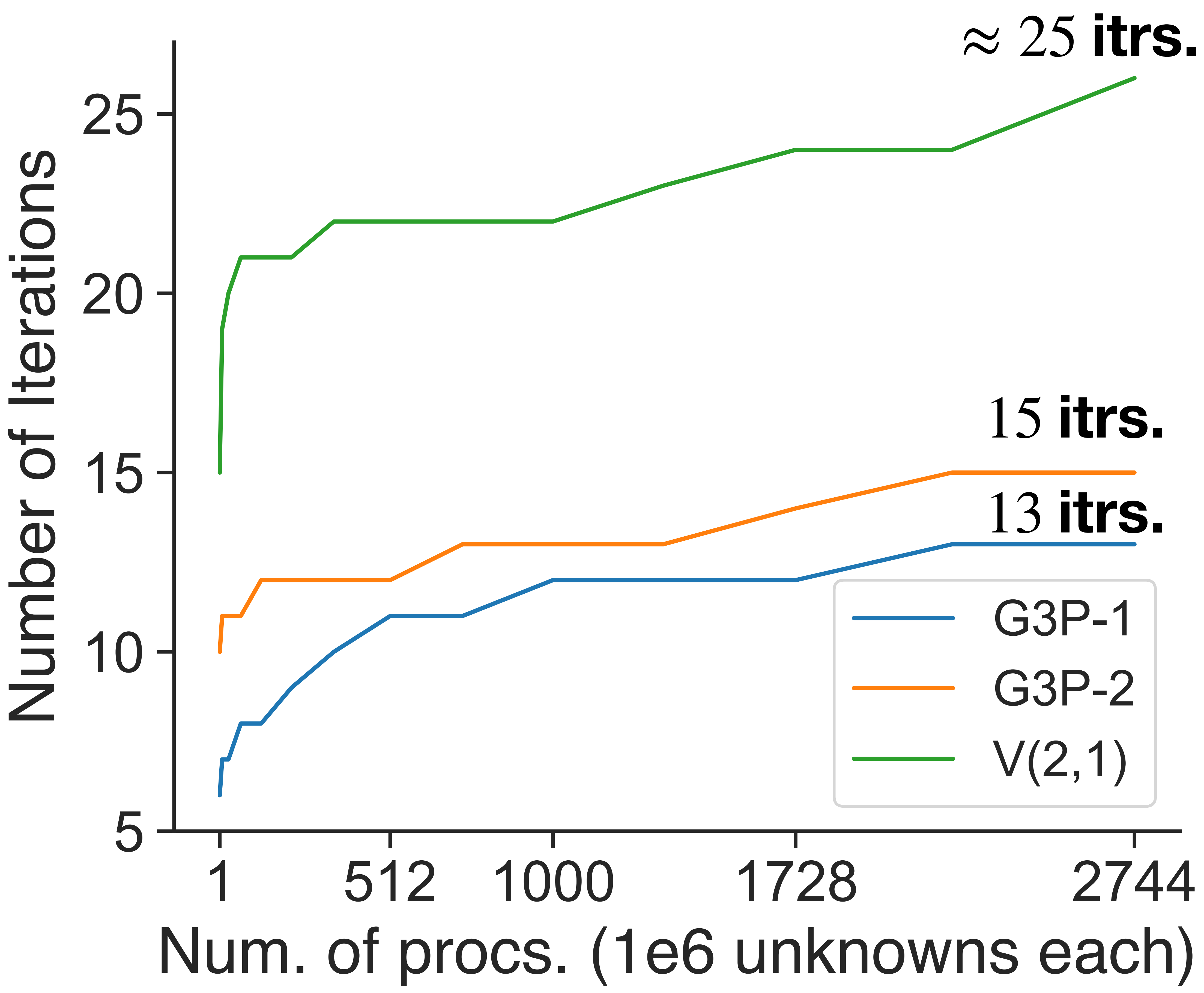
	(Solve Time (s), Number of Iterations) on a $100 \times 100 \times 100$ grid				
	f=0, ϵ=0.01	f=0, ϵ=0.001	f=0, ϵ=0.0001	f=1, ϵ=0.001	f=$rand$, ϵ=0.001
$V(2,1)$	(0.33, 10)	(0.33, 10)	(0.31, 10)	(0.32, 10)	(0.26, 8)
$V(3,2)$	(0.36, 9)	(0.31, 8)	(0.30, 8)	(0.31, 8)	(0.25, 6)
$V(3,3)$	(0.35, 8)	(0.34, 8)	(0.30, 7)	(0.31, 7)	(0.31, 6)
$G3P$-1	(0.27, 7)	(0.22, 6)	(0.22, 6)	(0.26, 7)	(0.19, 5)
$G3P$-2	(0.29, 10)	(0.28, 10)	(0.29, 10)	(0.29, 10)	(0.24, 8)

≈ 1.2**x** – 1.4**x**

≈ 1**x** – 1.2**x**

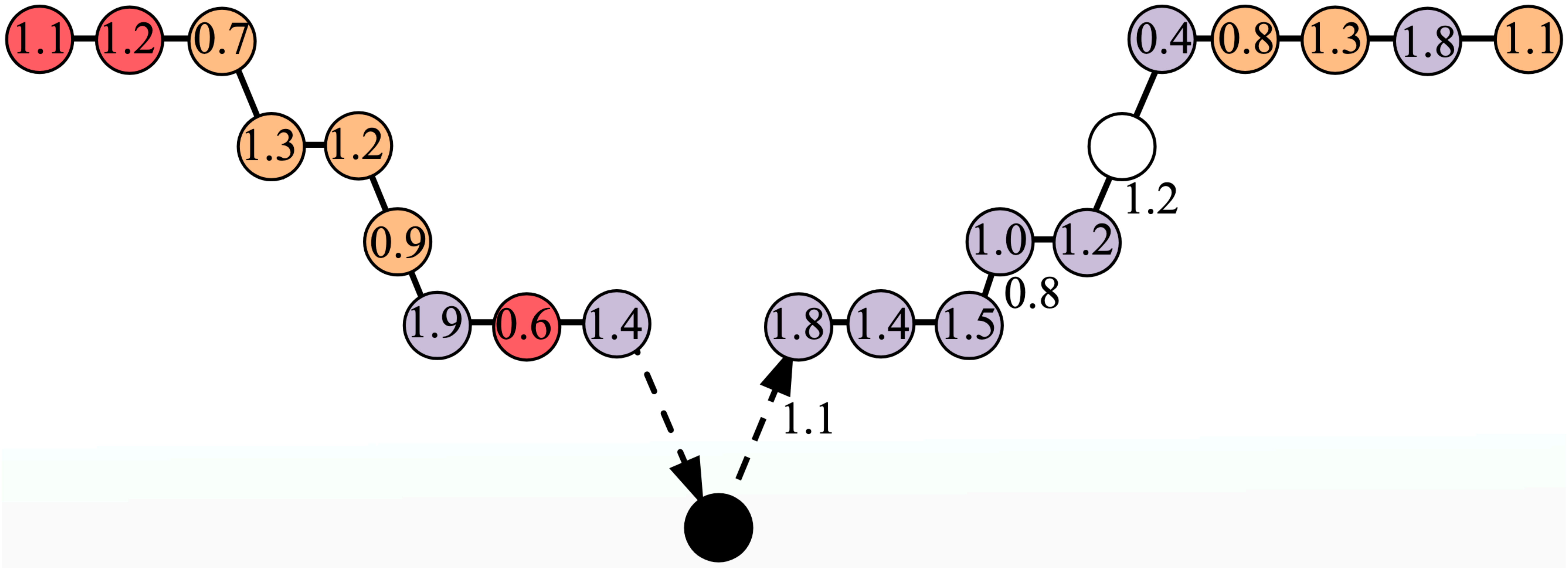
Weak scaling





Dinesh Parthasarathy
dinesh.parthasarathy@fau.de

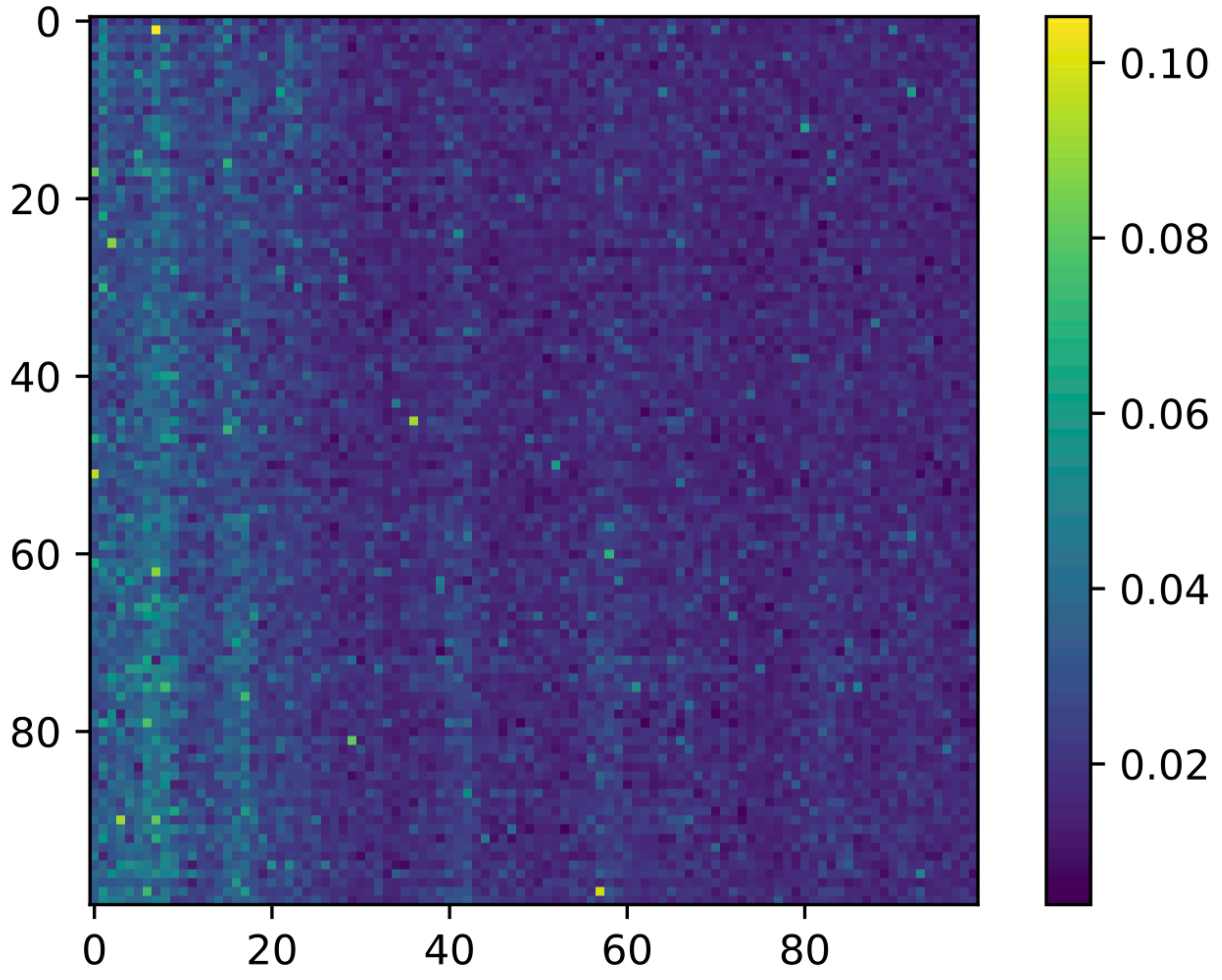
Damping of fourier modes

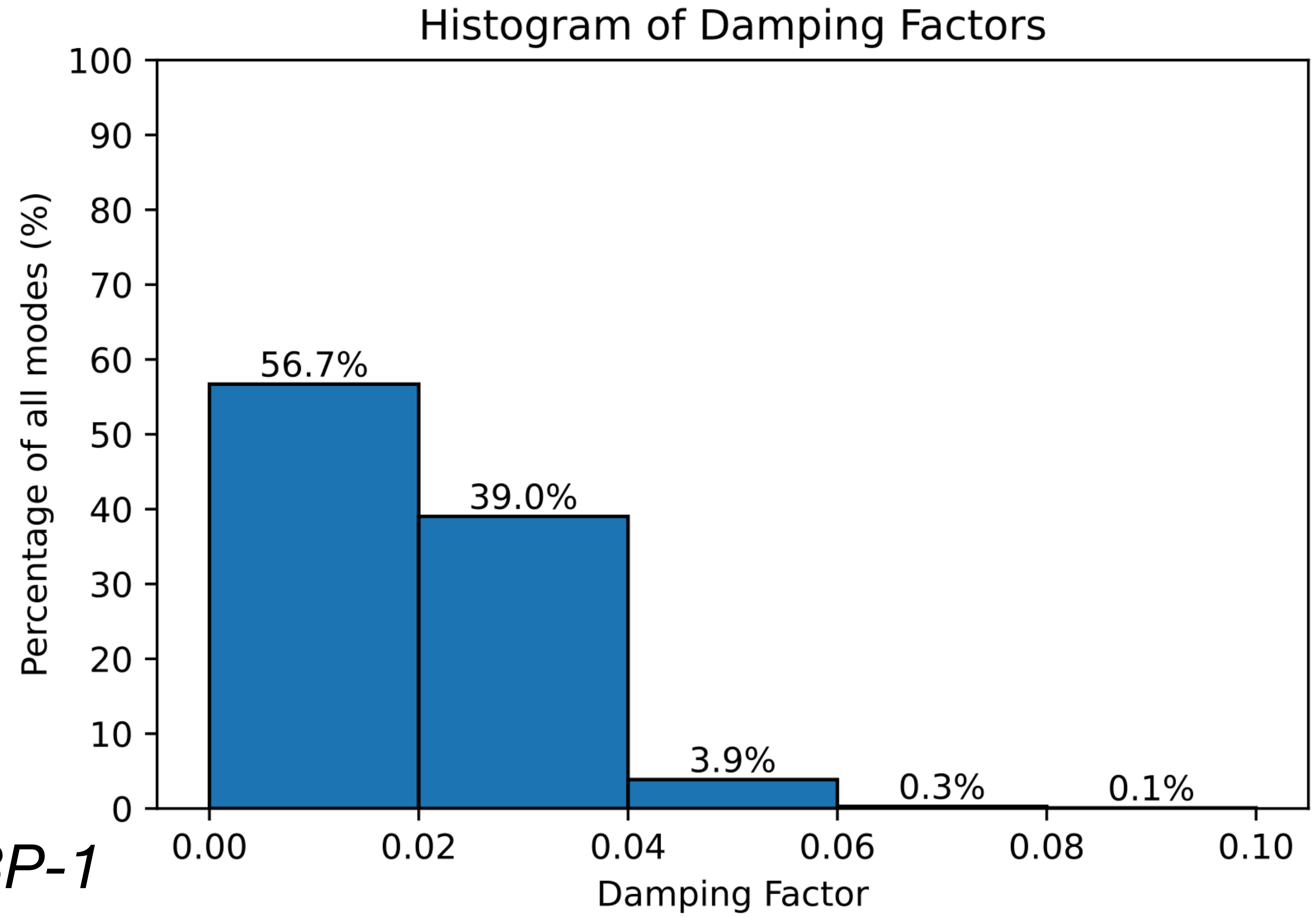


G3P-1

Dinesh Parthasarathy
dinesh.parthasarathy@fau.de

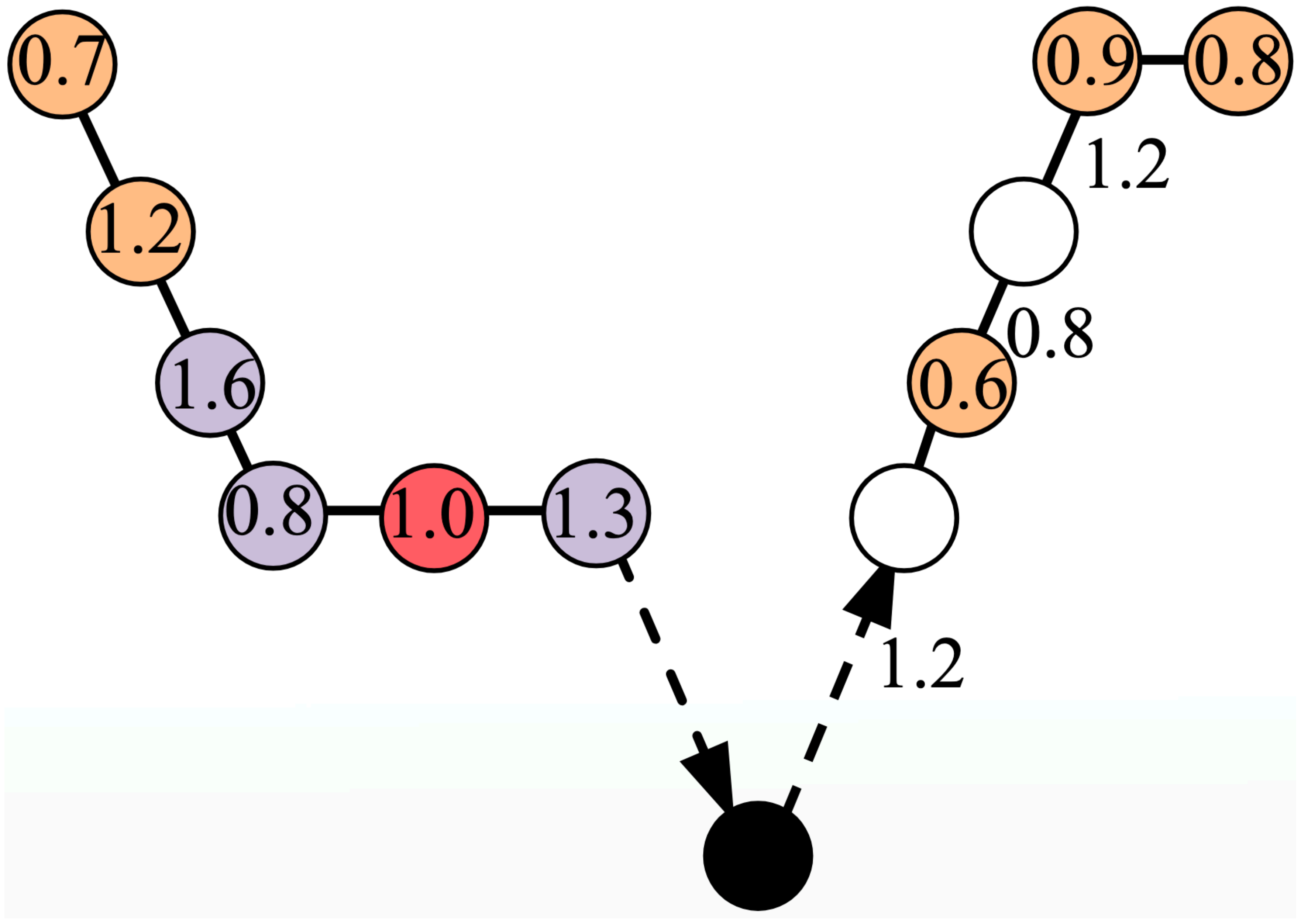
Damping of fourier modes





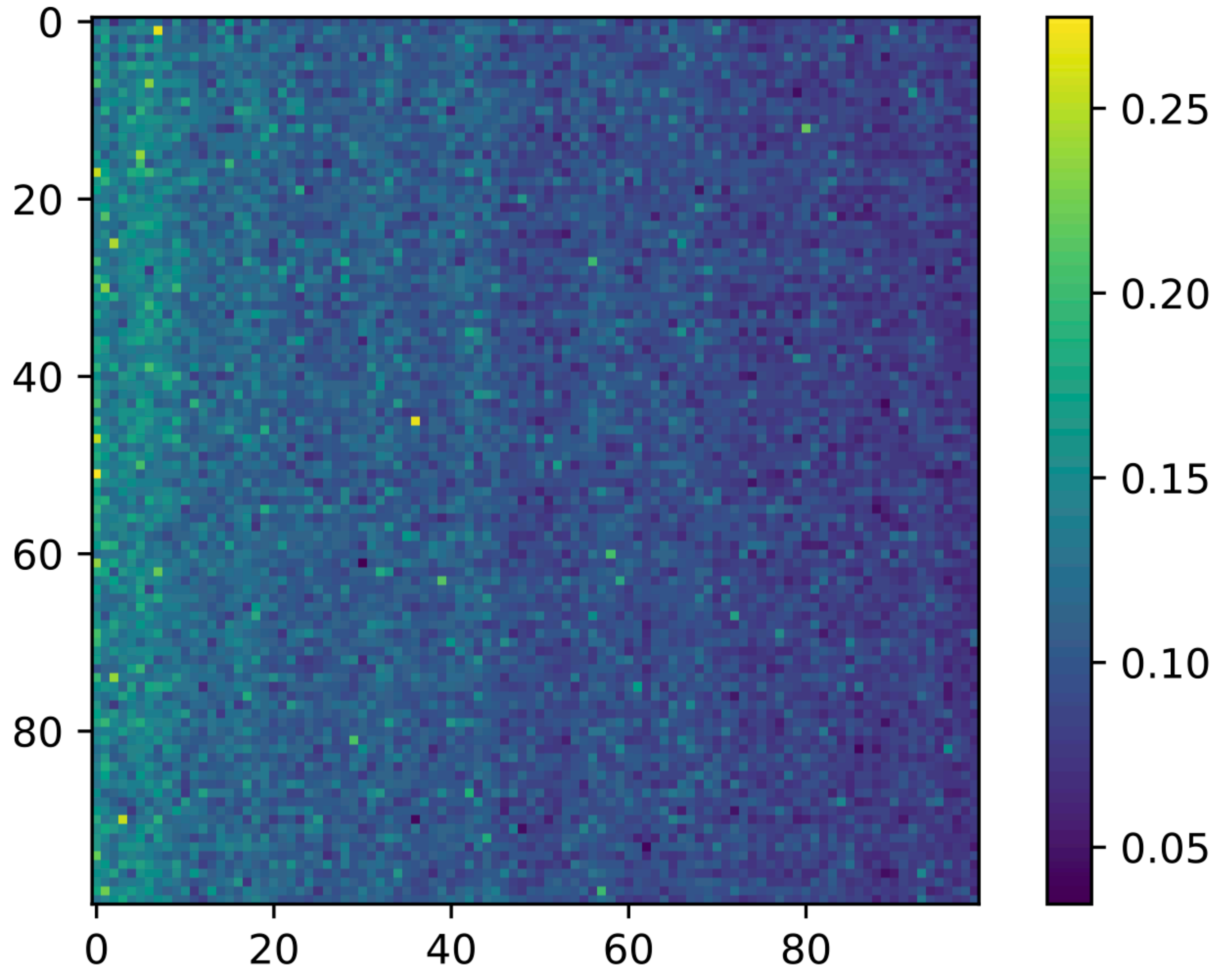
G3P-1

Damping of fourier modes

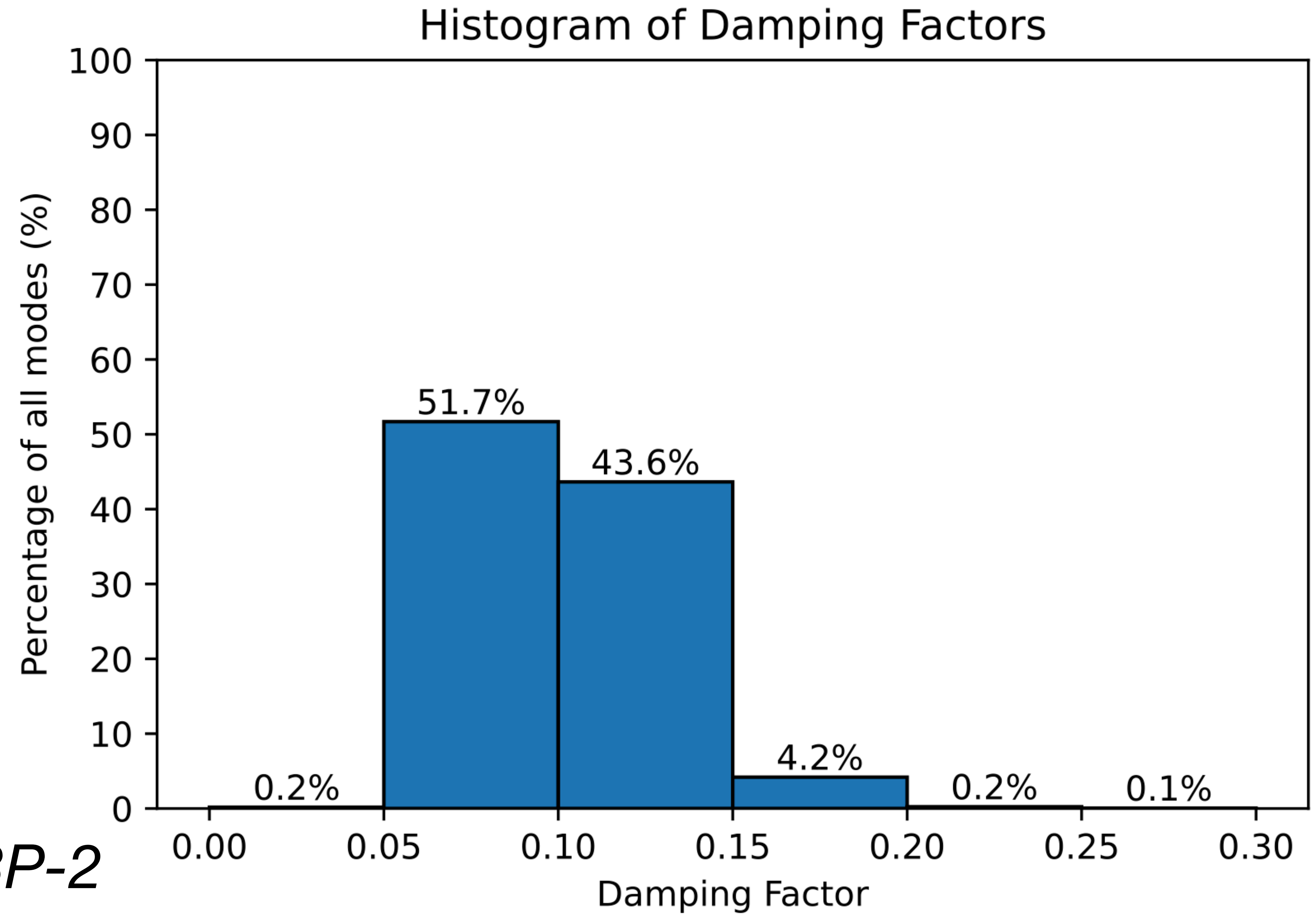


G3P-2

Damping of fourier modes



G3P-2



Time-stepping problem

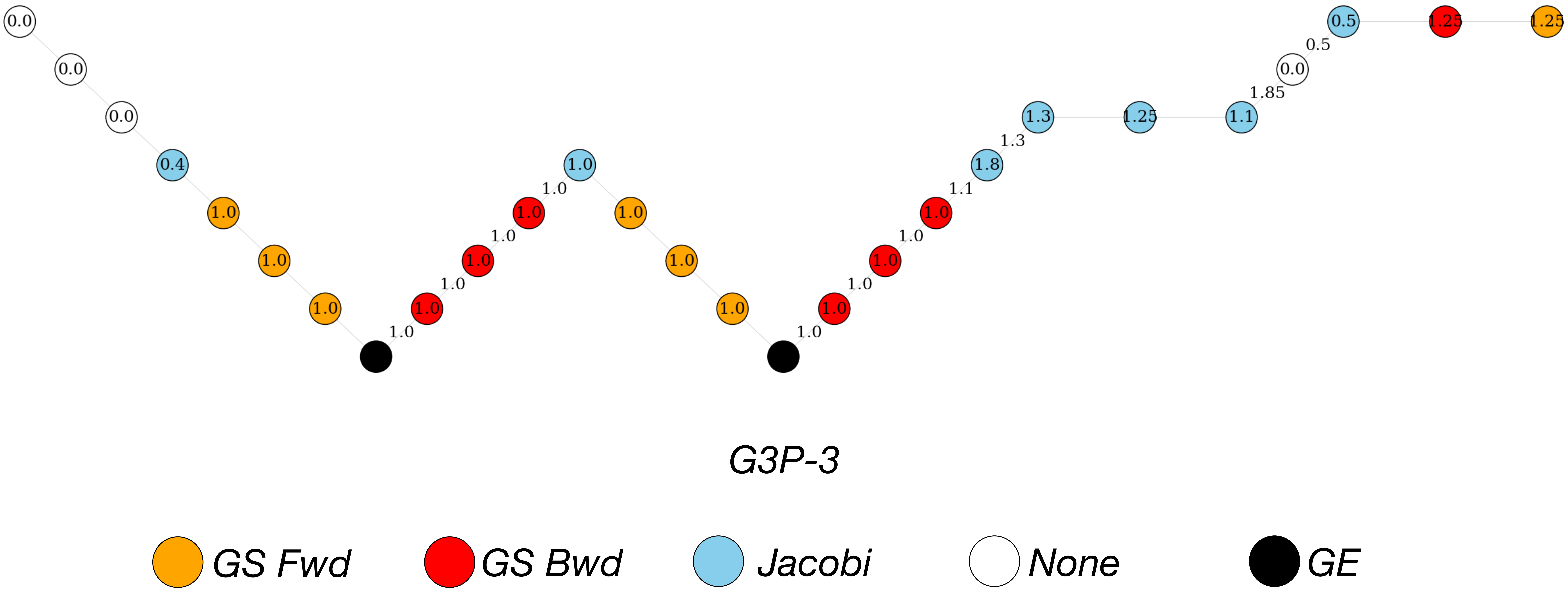
Problem setup

- 3D radiation-diffusion application on Ares, a multiphysics code developed at LLNL.

- AMG-PCG : 1 cycle of AMG per CG.

- Evolve AMG cycles for a system from a single timestep (512k unknowns, 8lvls.)

Preconditioner structure



Friedrich-Alexander-Universität Erlangen-Nürnberg
Dinesh Parthasarathy
dinesh.parthasarathy@fau.de
Lawrence Livermore National Laboratory

Results

	V(1,1)		G3P-3	
	T(ms)	CG Itrs.	T(ms)	CG Itrs.
t=1	330	15	**210**	10
t=2	270	12	**240**	11
t=3	290	13	**210**	10
t=4	290	12	**230**	11
t=5	310	14	**210**	10
t=6	240	10	**220**	10
t=7	300	13	**210**	10
t=8	220	9	**210**	10
t=9	320	14	**210**	10
t=10	220	9	**200**	9
	2790ms		**2150ms**	

≈ 1.3**x**

Conclusion

- Optimal flexible cycles perform better compared to standard cycle types.
- 3D anisotropic: generalises well over problem variants and sizes.
- Time-stepping code: generalises well over multiple timesteps, when the system matrix is perturbed.
- Produces fast solvers which are still interpretable (not a black-box algorithm).

Dinesh Parthasarathy
dinesh.parthasarathy@fau.de

Future directions

- Clever initialisation of starting population of solvers.
- Refine grammar during evolution.
- Extend the grammar rules : AMG setup, other solvers and preconditioners.
- Use the Pareto optimal solutions as data to train a ML model.

Dinesh Parthasarathy
dinesh.parthasarathy@fau.de

Thank you for listening!
Questions?

Dinesh Parthasarathy
dinesh.parthasarathy@fau.de

References

[1] R. Huang, R. Li, and Y. Xi, Learning optimal multigrid smoothers via neural networks, SIAM J. Sci. Comput., 45 (2023), pp. S199–S225.

[2] Luz, M. Galun, H. Maron, R. Basri, and I. Yavneh, Learning algebraic multigrid using graph neural networks, in Proceedings of the 37th International Conference on Machine Learning, ICML'20, JMLR.org, 2020.

[3] A. Taghibakhshi, S. P. MacLachlan, L. Olson, and M. West, Optimization-based algebraic multigrid coarsening using reinforcement learning, ArXiv, abs/2106.01854 (2021), https: //api.semanticscholar.org/CorpusID:235313621.

[4] Schmitt, J., Kuckuk, S. & Köstler, H. EvoStencils: a grammar-based genetic programming approach for constructing efficient geometric multigrid methods. *Genet Program Evolvable Mach* **22**, 511–537 (2021). https://doi.org/10.1007/s10710-021-09412-w

.