Optimising Multigrid Solvers Using Grammar-Guided Genetic Programming

Dinesh Parthasarathy
University of Erlangen-Nuremberg

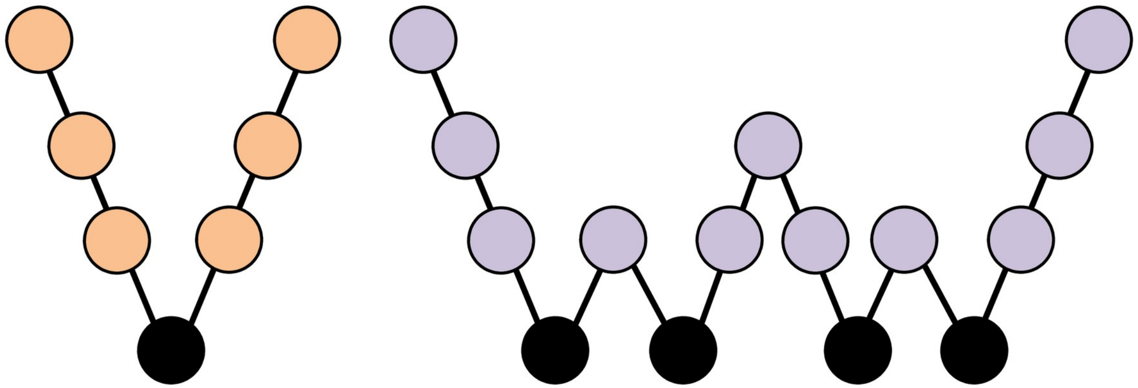
Centre of Excellence in Exascale CFD

Introduction

$$Ax = b$$

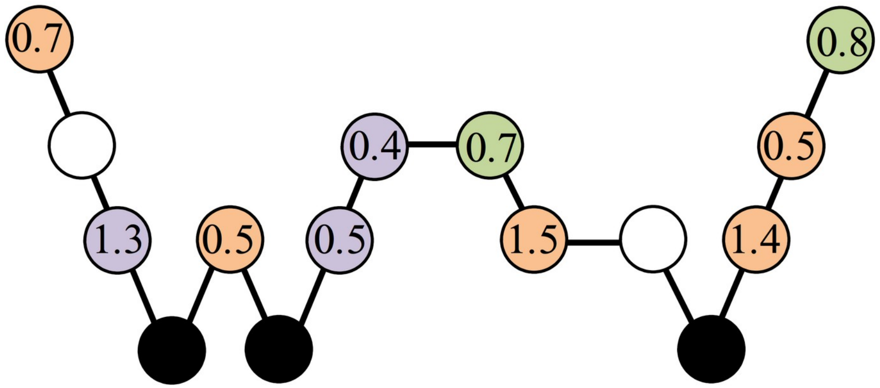
Linear systems of equations arising from many PDEs can be efficiently solved using multigrid methods.

Motivation



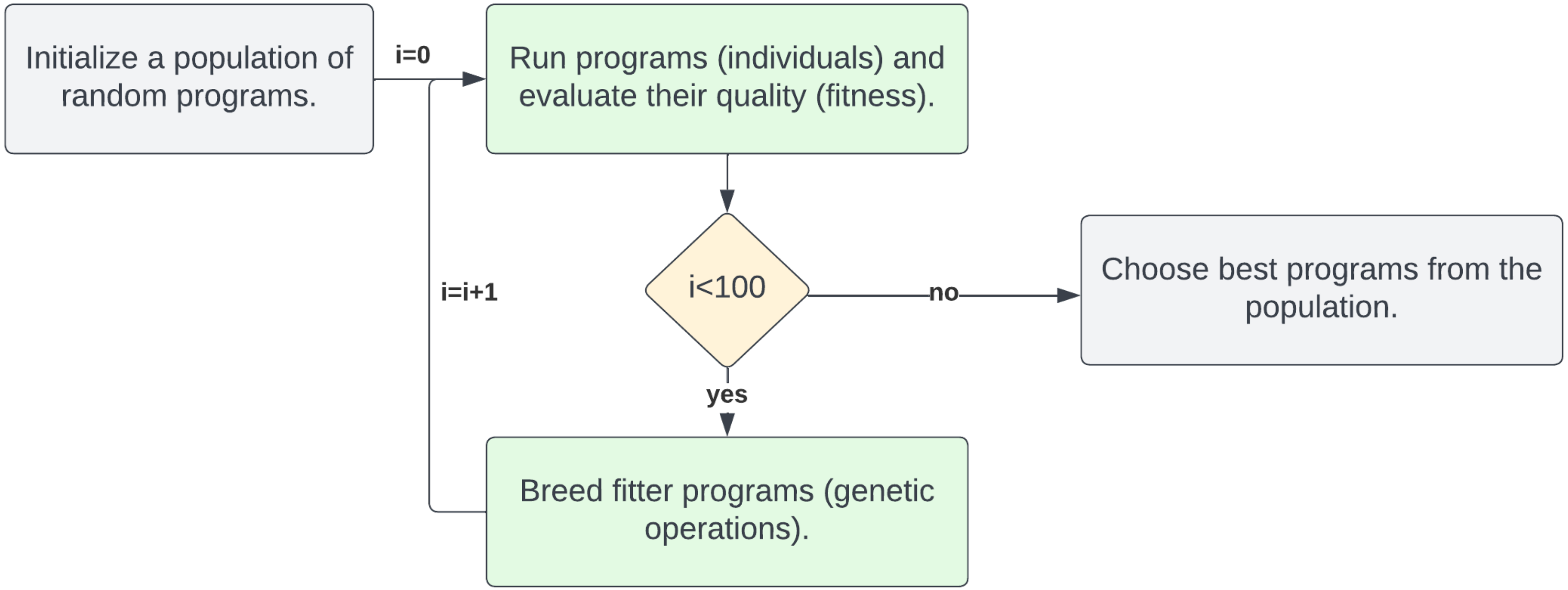
Standard Multigrid Cycles

Large search space !!



Flexible Multigrid Cycles

Genetic Programming – Primer

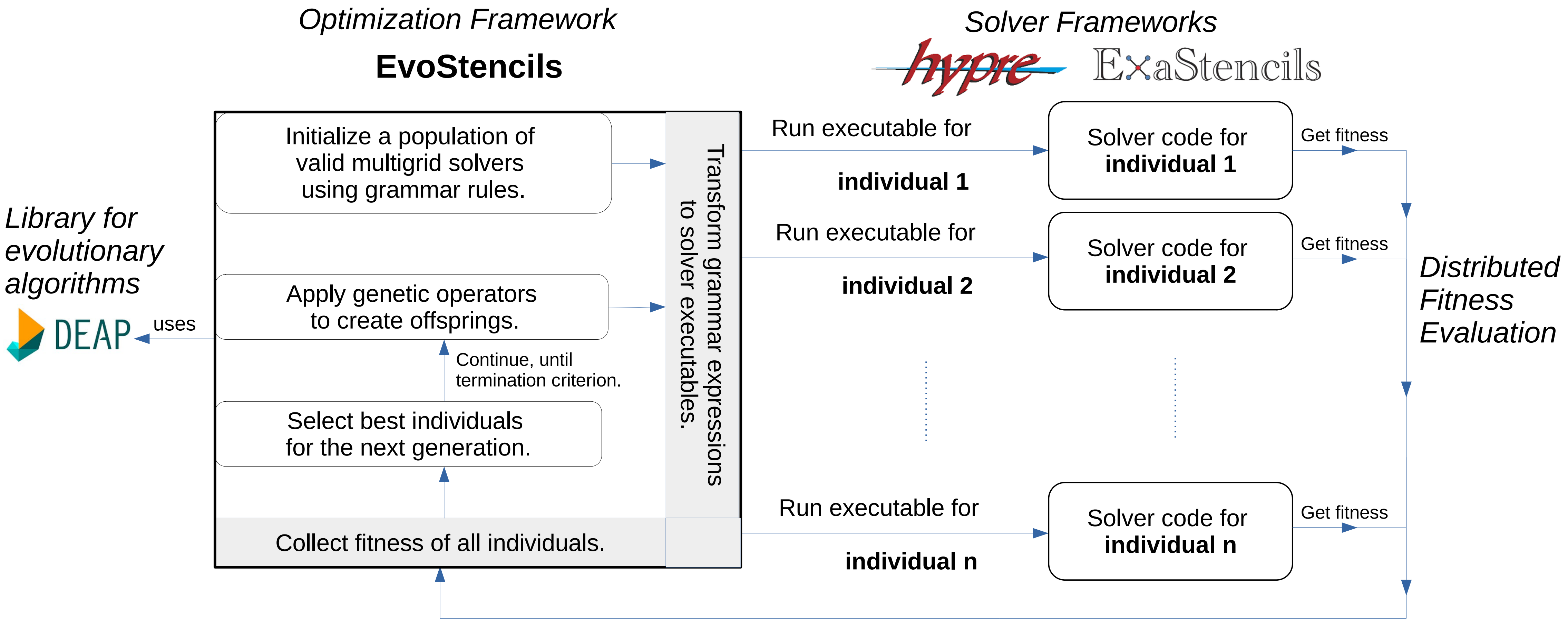


Genetic Programming – Grammar-Guided Approach[1]

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

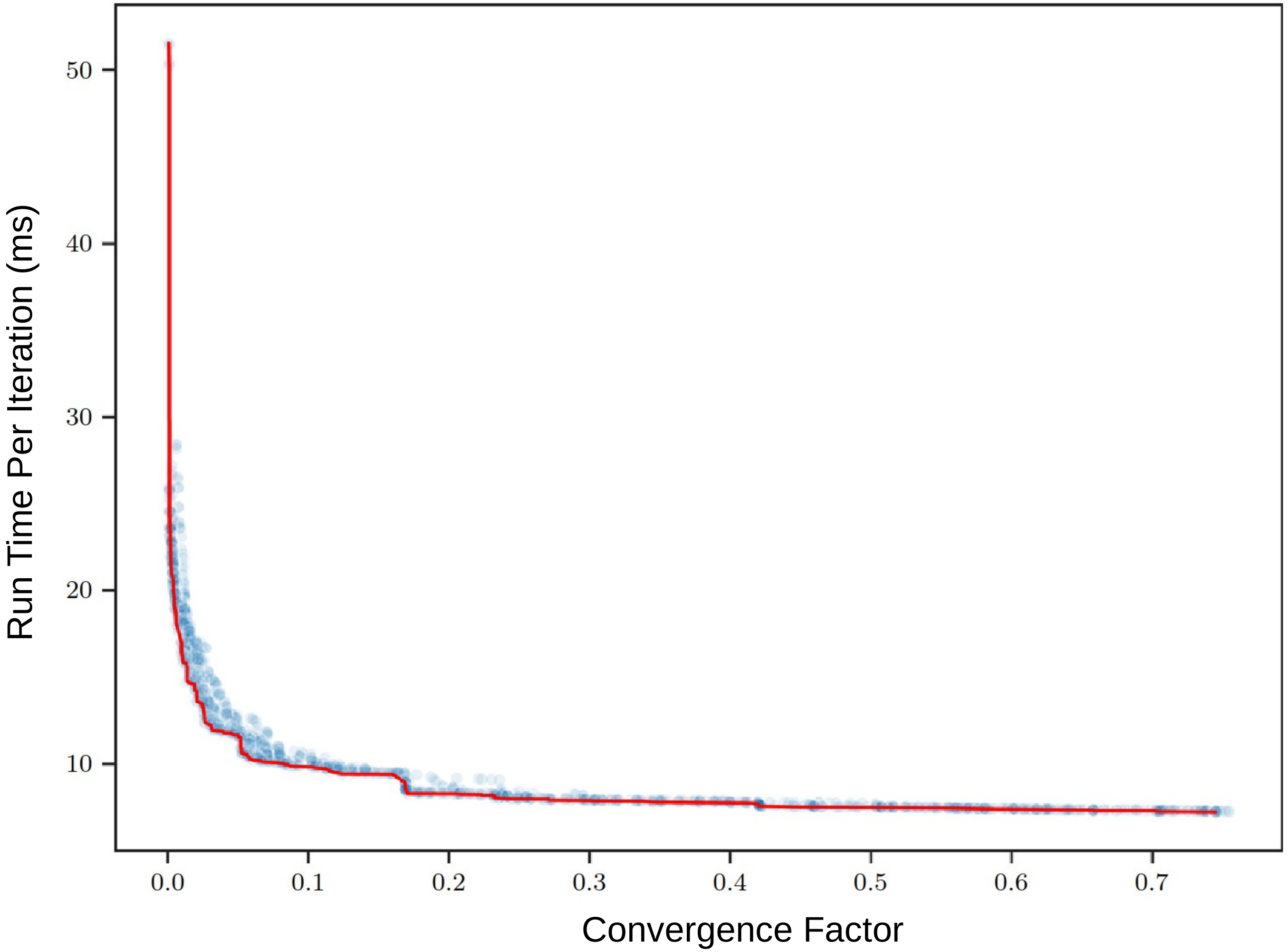
Production rules for constructing three-grid multigrid cycles where each symbol on the left side of the ⊨ sign can be replaced by the corresponding symbol on its right side.

Software Framework



Source code repository: https://github.com/jonas-schmitt/evostencils

Objectives



50
40
30
20
10
Run Time Per Iteration (ms)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Convergence Factor

Optimisation Components

Smoothers	Jacobi, Gauss-Seidel, RBGS, Jacobi-Newton, RBGS-Newton…., etc.
Restriction Operator	Full-weighting restriction
Prolongation Operator	Bilinear interpolation
Relaxation factors	(0.1, 0.15, 0.2, …, 1.9)
Coarse grid solver	Gauss Elimination / CG / Jacobi-Newton /…. / etc.

Results – Nonlinear Multigrid[2]

$$-\Delta u(x,y) + \gamma u(x,y) e^{u(x,y)} = f(x,y) \quad \text{in } \Omega,$$

$$u(x,y) = 0 \quad \text{on } \Omega$$

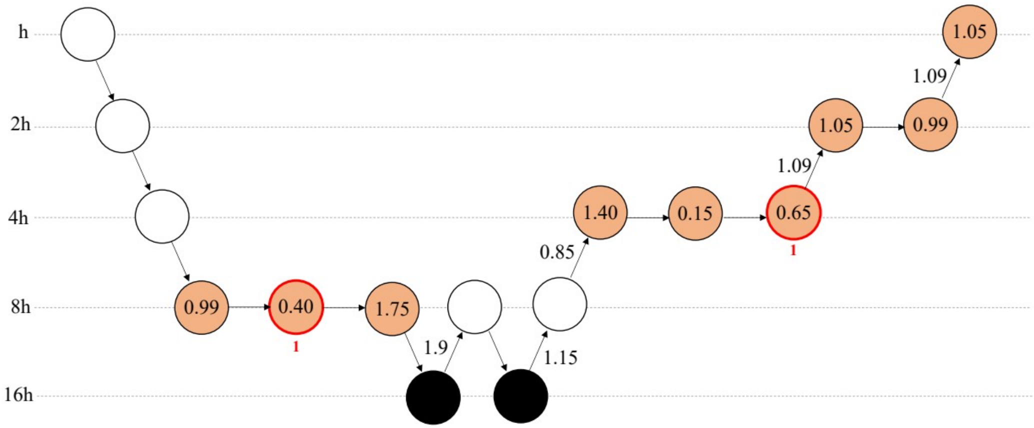
$$f(x,y) = \left((9\pi^2 + \gamma e^{(x^2-x^3) sin(3\pi y)}) \cdot (x^2 - x^3) + 6x - 2 \right) \cdot sin(3\pi y)$$

$$l_{max} = 10 \quad \gamma = 20$$

2D nonlinear problem.

	Runtime (ms)	n_Iterations	Convergence
V(1,1)	306.99	21	0.32
V(1,2)	238.56	14	0.18
V(2,1)	254.78	15	0.21
V(2,2)	253.72	13	0.17
W(2,2)	1017.18	9	0.07
GGGP-1	117.88	7	0.04
GGGP-2	115.09	8	0.05
GGGP-3	118.39	11	0.12

Comparing efficiency of Grammar-Guided Genetic Programming (GGGP / G3P) generated solvers with handcrafted reference methods.



The structure of the GGGP -3 solver . The color of the node denotes the type of operation. Brown: Red-Black Gauss-Seidel smoothing, Black: Coarse Grid Solver, White:No Operation. The borders of the node indicate the type of approximation used in the smoothing operation. Red: Newton's approximation with the number of Newton steps indicated in red text, Black: Picard's approximation. The relaxation factors are indicated inside the node for smoothing and on the edges for coarse-grid correction.

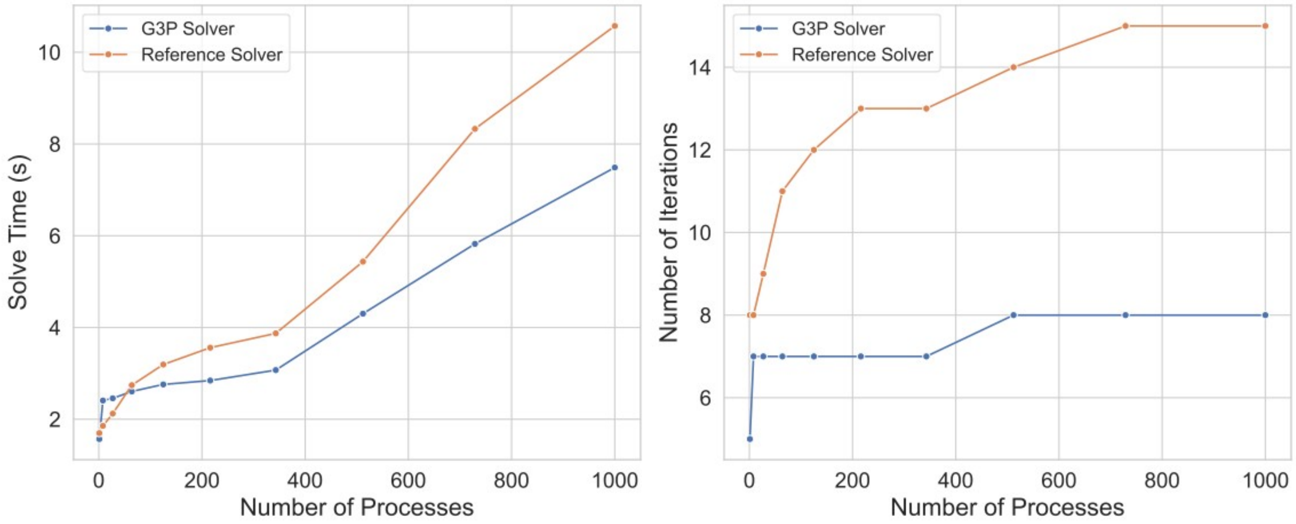
Results – Algebraic Multigrid

$$c_x \frac{\partial^2 \phi}{\partial x^2} + c_y \frac{\partial^2 \phi}{\partial y^2} + c_z \frac{\partial^2 \phi}{\partial z^2} = 0$$

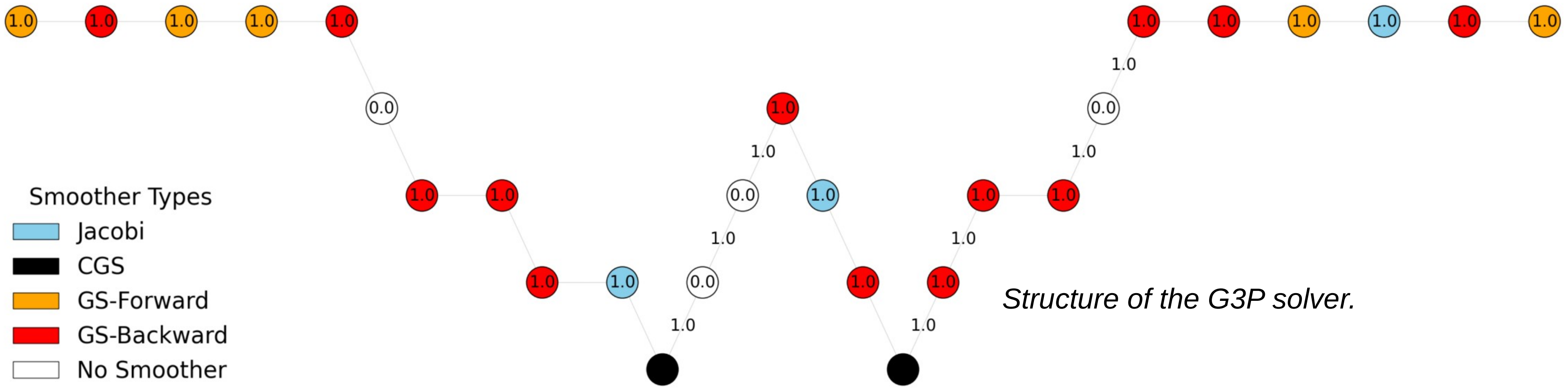
$$c_x = 0.001\,,\quad c_y = 1\,,\quad c_z = 1$$

$$N_x = 100\,,\ N_y = 100\,,\ N_z = 100$$

3D anistropic poisson problem.



Weak scaling of the G3P solver and an optimised V-cycle solver.

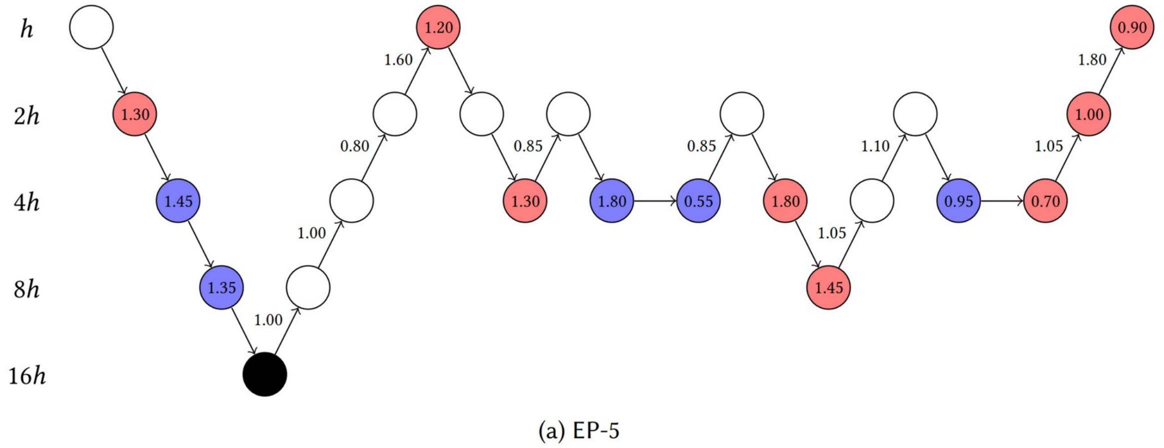


Structure of the G3P solver.

Results – Multigrid Preconditioner [3]

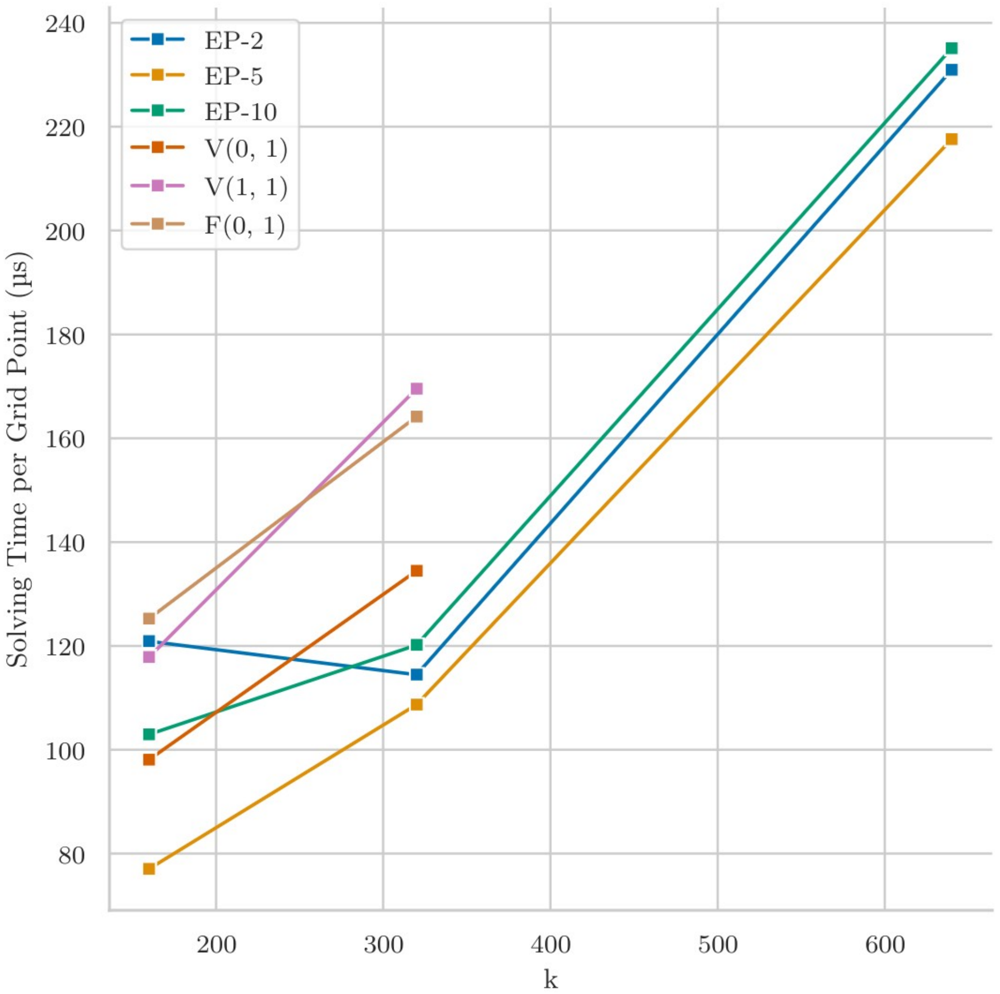
$$(-\nabla^2 - k^2)u = f \quad \text{in } (0,1)^2$$
$$u = 0 \quad \text{on } (0,1)\times\{0\}, (0,1)\times\{1\}$$
$$\partial_{\mathbf{n}} u - iku = 0 \quad \text{on } \{0\}\times(0,1), \{1\}\times(0,1)$$
$$f(x,y) = \delta(x-0.5, y-0.5),$$

2D indefinite Helmholtz problem.



(a) EP-5

Computational structure of the evolved multigrid preconditioners. The color of the node denotes the type of operation. Black:Coarse-grid solver, Blue: Pointwise Jacobi smoothing, Red: Red-black Gauss-Seidel smoothing, White: No operation. The relaxationfactor of each smoothing step is included in each node, while for coarse-grid correction, it is attached to the respective edge.



Solving time comparison of the best preconditioners.

References

[1] 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

[2] Dinesh Parthasarathy, Jonas Schmitt, and Harald Köstler. 2023. Evolving Nonlinear Multigrid Methods With Grammar-Guided Genetic Programming. In Proceedings of the Companion Conference on Genetic and Evolutionary Computation (GECCO '23 Companion). Association for Computing Machinery, New York, NY, USA, 615–618. https://doi.org/10.1145/3583133.3590734

[3] Jonas Schmitt and Harald Köstler. 2022. Evolving generalizable multigrid-based helmholtz preconditioners with grammar-guided genetic programming. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '22). Association for Computing Machinery, New York, NY, USA, 1009–1018. https://doi.org/10.1145/3512290.3528688

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Sweden, Germany, Spain, Greece, and Denmark under grant agreement No 101093393.

Co-funded by the European Union

EuroHPC Joint Undertaking

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European High Performance Computing Joint Undertaking (JU) and Sweden, Germany, Spain, Greece, and Denmark. Neither the European Union nor the granting authority can be held responsible for them.

CEEC

Centre of Excellence in Exascale CFD

Thank you
for your attention!

CEEC
Centre of Excellence in Exascale CFD