14th World Congress on Computational Mechanics (WCCM)
ECCOMAS Congress 2020)
Virtual Congress: 11-15 January 2021
F. Chinesta, R. Abgrall, O. Allix and M. Kaliske (Eds)

CHOOSING THE SUBREGIONS IN THREE-LEVEL FROSCH PRECONDITIONERS

Alexander Heinlein[1], Oliver Rheinbach[2,3] and Friederike Röver[2]

[1] Institute of Applied Analysis and Numerical Simulation, Universität Stuttgart,
Pfaffenwaldring 57, 70569 Stuttgart, Germany.
E-mail: Alexander.Heinlein@ians.uni-stuttgart.de

[2] Institut für Numerische Mathematik und Optimierung, Technische Universität Bergakademie Freiberg
Akademiestr. 6, 09599 Freiberg, Germany.

[3] Universitätsrechenzentrum (URZ), Technische Universität Bergakademie Freiberg
Akademiestr. 6, 09599 Freiberg, Germany.

Key words: Domain Decomposition, Parallel Computing, Overlapping Schwarz, Mesh Partitioning, Coarse Operator, Preconditioners, Trilinos

Abstract. Different graph partitioning methods, i.e., linear partioning, parallel hypergraph (PHG) partioning, and two approaches using ParMETIS, are considered to generate an unstructured decomposition of the second-level coarse operator of three-level FROSch (Fast and Robust Overlapping Schwarz) preconditioners in the Trilinos software library. In our context, the parallel hypergraph method shows the most consistent results.

1 Three-Level FROSch Preconditioners

FROSch (Fast and Robust Overlapping Schwarz) [7] is a parallel implementation of the Schwarz framework and is part of the software library Trilinos [1]. It provides parallel implementations of the GDSW (Generalized Dryja–Smith–Widlund) [5, 4] and RGDSW (Reduced-dimension GDSW) [6] overlapping Schwarz preconditioners, which make use of exotic coarse spaces. We consider the extension of these methods to three-level (R)GDSW preconditioners [9, 10], which arise from the recursive application of the two-level methods. They are therefore based on a hierarchy of decompositions of the original domain Ω; the domain is decomposed into non-overlapping subregions $\{\Omega_{i0}\}_{i=1,\dots,N_0}$, which are further decomposed into non-overlapping subdomains $\{\Omega_i\}_{i=1,\dots,N}$. Here, in order to build the subdomain hierarchy, we proceed in reverse order: We first partition Ω into the non-overlapping subdomains. Then, we build the dual graph of this decomposition based on the connectivity and obtain the subregions by partitioning this graph. The non-overlapping subdomains are then extended by layers of elements, resulting in overlapping subdomains $\{\Omega_i'\}_{i=1,\dots,N}$ with overlap δ, and the subregions are extended by layers of subdomains to obtain the overlapping subregions $\{\Omega_{i0}'\}_{i=1,\dots,N_0}$ with overlap Δ.

The three-level (R)GDSW overlapping Schwarz preconditioners are then of the form

$$M_{\mathrm{GDSW-3L}}^{-1} = \Phi\Big(\overbrace{\Phi_0 K_{00}^{-1}\Phi_0^T}^{\text{Third Level}} + \overbrace{\textstyle\sum_{i=1}^{N_0} R_{i0}^T K_{i0}^{-1} R_{i0}}^{\text{Second Level}}\Big)\Phi^T + \overbrace{\textstyle\sum_{j=1}^{N} R_j^T K_j^{-1} R_j}^{\text{First Level}}, \tag{1}$$

where the restriction operators R_i to the overlapping subdomains and the respective matrices $K_i = R_i K R_i^T$ as well as the (R)GDSW basis functions Φ and the matrix $K_0 = \Phi^T K \Phi$ are defined as in the two-level methods; see, e.g., [4]. Furthermore, restriction operators $R_{i0} : V^0 \to V_i^0 := V^0(\Omega'_{i0})$, corresponding to the overlapping subregions Ω'_{i0}, as well as coarse basis functions Φ_0, spanning the (R)GDSW coarse space V_{00} on the third level, are needed to build the three-level preconditioners; see [9, 10, 8] for more details. Note that the three-level (R)GDSW approach is related to three-level or multilevel BDDC methods [16, 14, 2, 15]. Our undertaking is, software-wise, of course related to other important scalable implementations of Schwarz methods, e.g., [12].

2 Implementation

The three-level extension of GDSW type preconditioners has recently been added to the FROSch framework; see also [8]. The implementation is based on the Trilinos package Xpetra, which provides a lightweight interface to the parallel linear algebra packages Epetra and Tpetra. Here, we only use the newer Tpetra-based software stack. For the assembly of the system matrix, we employ the finite difference implementation based on a structured tensor product mesh and a structured domain decomposition into rectangular subdomains from the Galeri package. In order to partition the dual graph corresponding to the decomposition into the subdomains $\{\Omega_{i0}\}_{i=1,\dots,N_0}$, we use the Trilinos package Zoltan2 [17]. It provides an interface to the partitioning algorithms from the older Zoltan package and can also be linked to third party libraries such as ParMETIS [13]. We solve the linear system assembled by Galeri using the conjugate gradient (PCG) method from the Trilinos package Belos, preconditioned by FROSch. Both the iterative solver and the three-level FROSch preconditioner are called using the unified solver interface Stratimikos, and we iterate until we reach relative tolerance of 10^{-6}. All tests were performed on the Compute Cluster of the Fakultät für Mathematik und Informatik of the TU Freiberg. We use one subdomain for each MPI rank, and one MPI rank for each processor core. We use the INTEL compiler 2020.0 version 3.1.4. To solve the arising subproblems, we always use the PardisoMKL [3] sparse direct solver.

3 Results

In this section, we will investigate numerically how different graph partitioning algorithms employed to construct the nonoverlapping subregions $\{\Omega'_{i0}\}_{i=1,\dots,N_0}$ influence the performance of three-level FROSch preconditioners. As a model problem, we consider linear elasticity with homogeneous Dirichlet boundary conditions on the unit square $[0,1]^2$ or unit cube $[0,1]^3$, respectively. As mentioned before, we use finite differences as the spatial discretization. To build the subregions, different partitioning methods are compared: linear partitioning, which is denoted block-wise partitioning in Zoltan (*Block*), the parallel hyper graph (PHG) partitioner from Zoltan, and two partitioning approaches based on ParMETIS. In particular, for ParMETIS, we use either *partitioning from scratch (P)* or *repartitioning (R)* of the initial partition. Here, the initial partition is the linear partition. We used ParMETIS Version 4.0.3. We present results for the GDSW as well as for the RGDSW coarse space, where, for the RGDSW coarse space,

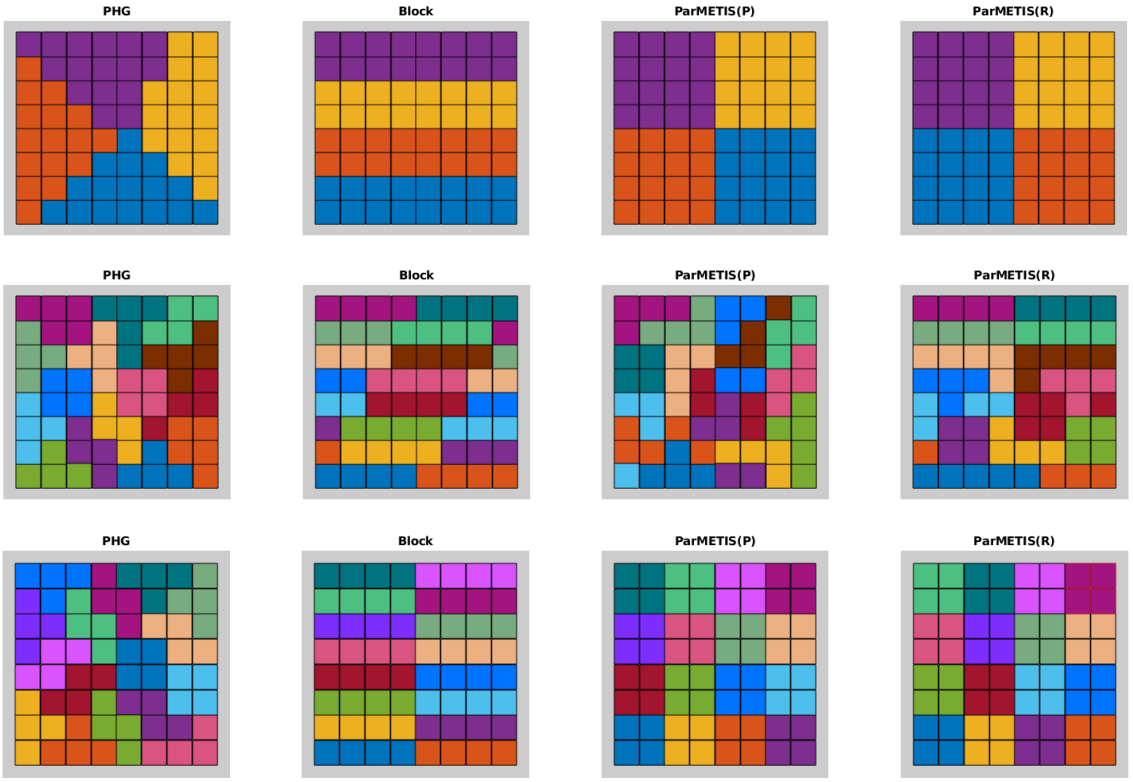

Figure 1: Decomposition of 64 subdomains into 4 (top), 15 (middle) and 16 (bottom) subregions using the parallel hypergraph (*PHG*), block-wise (*Block*), and ParMETIS approach. ParMETIS (P) uses partitioning from scratch and ParMETIS (R) uses repartitioning of the initial distribution, which is the linear distribution. Each square corresponds to one subdomain and each color to one subregion. This partitioning determines the parallel distribution of the second-level coarse operator K_0 in FROSch and results in a different third-level coarse matrix K_{00}.

we always consider *Option 1* from [6, 11], which is completely algebraic. Note that, since the subregions are built based on the dual graph of the domain decomposition, the subregions are identical for the same configuration independent of the choice of the coarse space. However, the size of the GDSW and RGDSW coarse spaces may still differ significantly. In order to investigate the influence of the different partitioning algorithms on the three-level preconditioners, we focus on two aspects. First, we investigate how the convergence, i.e., the number of iterations and the estimated condition number, depend on the partitioning. Second, we will discuss the dimension of the coarse space on the third level V_{00} and thus the size of the coarse matrix K_{00}; cf. (1). In order to do so, we consider a structured domain decomposition into 64 or 512 subdomains for two or three dimensions, respectively, and vary the number of subregions.

		Three-level GDSW preconditioner				Three-level RGDSW preconditioner			
		PHG	Block	ParMETIS(P)	ParMETIS(R)	PHG	Block	ParMETIS(P)	ParMETIS(R)
2D; #subdomains = 64									
#subreg.		dim(K_0) = 483				dim(K_0) = 147			
4	iter	35	33	34	**32**	35	**33**	44	37
	$\kappa(M^{-1}K)$	19.7	24.8	19.6	**17.6**	17.5	**15.1**	30.7	21.7
	dim(K_{00})	21	**9**	15	15	6	9	**3**	**3**
	dim(K_{00})/#regs	5.3	2.3	3.8	3.8	1.5	2.3	0.8	0.8
15	iter	**36**	40	44	40	**43**	**43**	54	**43**
	$\kappa(M^{-1}K)$	**21.9**	32.5	30.8	29.2	26.5	26.3	43.1	**25.8**
	dim(K_{00})	147	**129**	144	132	54	**30**	45	45
	dim(K_{00})/#regs	9.8	8.6	9.6	8.8	3.6	2.0	3.0	3.0
16	iter	36	40	39	**32**	43	44	50	**40**
	$\kappa(M^{-1}K)$	21.4	33.1	24.8	**21.2**	26.5	27.1	40.3	**24.8**
	dim(K_{00})	147	**87**	99	99	51	**21**	27	27
	dim(K_{00})/#regs	9.2	5.4	6.2	6.2	3.2	1.3	1.7	1.7
3D; #subdomains = 512									
#subreg.		dim(K_0) = 17 178				dim(K_0) = 2 058			
4	iter	**48**	51	49	52	49	**44**	52	54
	$\kappa(M^{-1}K)$	**35.6**	51.2	38.5	41.2	35.1	**29.2**	39.7	42.1
	dim(K_{00})	66	**18**	30	66	**6**	18	**6**	**6**
	dim(K_{00})/#regs	16.5	4.5	7.5	16.5	1.5	4.5	1.5	1.5
16	iter	**59**	**59**	67	67	57	**54**	61	61
	$\kappa(M^{-1}K)$	**55.4**	59.7	71.2	71.4	45.5	**40.6**	52.6	52.0
	dim(K_{00})	1 056	**174**	402	402	180	**42**	**42**	**42**
	dim(K_{00})/#regs	66	10.9	21.1	21.1	11.3	2.6	2.6	2.6
45	iter	60	**58**	60	60	67	**60**	72	**60**
	$\kappa(M^{-1}K)$	65.6	**50.3**	59.6	55.2	45.5	**40.6**	52.6	52.0
	dim(K_{00})	3 210	2 532	7 656	**2 166**	540	504	1 146	**342**
	dim(K_{00})/#regs	71.3	56.3	170.1	47.0	12.0	11.2	25.5	7.6
64	iter	61	**56**	62	62	71	**63**	67	67
	$\kappa(M^{-1}K)$	63.4	**55.7**	93.7	93.7	74.1	**57.2**	69.8	69.8
	dim(K_{00})	4 998	**966**	1 674	1 674	852	294	**162**	**162**
	dim(K_{00})/#regs	78.1	15.1	26.2	26.2	13.3	4.6	2.5	2.5
101	iter	66	**61**	65	64	79	**77**	80	**77**
	$\kappa(M^{-1}K)$	68.9	**62.1**	68.6	64.7	87.8	**84.1**	93.6	84.2
	dim(K_{00})	**7 188**	7 716	9 564	8 328	**1 176**	1 290	1 542	1 572
	dim(K_{00})/#regs	71.2	76.4	94.7	82.5	11.6	12.8	15.3	15.6

Table 1: Number of PCG iterations *Iter*, condition number estimate $\kappa(M^{-1}K)$ and dimension of the coarsest problem dim(K_{00}) for the three-level extension with different number of subregions and partitioning methods. We have $H/h = 10$, $\delta = 1$ and $\Delta = 1$. In each row, the best results for the three-level GDSW and the RGDSW preconditioner are marked in **bold**.

Robustness The different partitioning methods result in significantly different decompositions into subregions; see Figure 1. This is the case even for a small number of subregions. On the other hand, our algorithm results in similar numbers of iterations and condition number estimates for all partitioning methods. We observe a slight increase in the number of iterations for an increasing number of subregions. The effect is more visible for RGDSW; see Table 1.

Partitioning The different partitioning schemes result in different numbers of interface components, i.e., faces, edges, and vertices. As a result, the size of the coarse matrix K_{00} may differ significantly. In certain cases, we observe unconnected subdregions, which increases the number of interface components

and therefore increases $dim(K_{00})$; also see Figure 1. In our experiments, ParMETIS sometimes produces a structured decomposition; see the cases of 4 and 16 subregions in Figure 1. In these cases, $\dim(K_{00})$ is small: using 16 subregions instead of 15 subregions in two dimensions reduces $dim(K_{00})$ to about 70% for ParMETIS as well as for blockwise partitioning; see Table 1. For PHG, we do not see this effect when changing the number of subregions from 15 to 16: the size of $dim(K_{00})$ stays the same for GDSW and reduces only slightly for RGDSW. The results for the test case of 512 subdomains, using 45 and 64 subregions in three dimensions, is consistent with this observation. For 45 subregions, the size of the K_{00} is significantly larger than for 64 subregions using ParMETIS (R), although we would expect the contrary. Here, ParMETIS with partitioning from scratch results in $dim(K_{00}) = 7\,656$ for the GDSW coarse space using 45 subregions. This compares to $dim(K_{00}) = 1\,674$ for 64 subregions. For the RGDSW coarse space $dim(K_{00})$ decreases by a factor more than 7: we have $dim(K_{00}) = 1\,146$ (45 subregions) and $dim(K_{00}) = 162$ (64 subregions); see Table 1. The PHG and the block approach show more consistent results with respect to $dim(K_{00})$. Note that ParMETIS (R), which uses repartitioning of the linear partition, often yields a smaller $dim(K_{00})$ compared to ParMETIS (P).

In total, the parallel hyper graph (PHG) method from *Zoltan*, which avoids unconnected subregions, as well as the block-wise approach, avoid outliers with very large $\dim(K_{00})$ present in the other methods.

Conclusion Concerning the different graph partitioning method to generate the subregions, our algorithm is robust in the condition number and the number of iterations. However, to keep the cost of the sparse direct linear coarse solver low, we want to keep $dim(K_{00})$ small. For certain special cases, ParMETIS produces a structured decomposition resulting in a very small $dim(K_{00})$. However, changing the numbers of subregions slightly can result in a very large $dim(K_{00})$ for ParMETIS (P). Here, repartitioning a linear decomposition using ParMETIS (R) can be a better choice. Although the results for the block-wise partitioning show good results, we recommend the PHG partitioning method. We expect that, for larger problems, the block-wise approach and, to a lesser extend, also ParMETIS (R) (using the block-wise approach as initial partition) will suffer from elongated subregions, which will then result in slower Krylov convergence.

Acknowledgements

The author acknowledge computing time on the Compute Cluster of the Fakultät für Mathematik und Informatik of Technische Universität Freiberg (DFG project number 397252409), operated by the university computing center (URZ). The second and third author would like to acknowledge funding by the Deutsche Forschungsgemeinschaft (DFG) under the DFG project number 441509557 within the DFG SPP 2256.

REFERENCES

[1] Trilinos public git repository. Web, 2018. https://github.com/trilinos/trilinos.

[2] S. Badia, A. F. Martín, and J. Principe. Multilevel balancing domain decomposition at extreme scales. *SIAM J. Sci. Comput.*, 38(1):C22–C52, 2016.

[3] Matthias Bollhöfer, Olaf Schenk, Radim Janalik, Steve Hamm, and Kiran Gullapalli. *State-of-the-Art Sparse Direct Solvers*, pages 3–33. Springer International Publishing, Cham, 2020.

[4] C. R. Dohrmann, A. Klawonn, and O. B. Widlund. Domain decomposition for less regular subdomains: overlapping Schwarz in two dimensions. *SIAM J. Numer. Anal.*, 46(4):2153–2168, 2008.

[5] C. R. Dohrmann, A. Klawonn, and O. B. Widlund. A family of energy minimizing coarse spaces for overlapping Schwarz preconditioners. In *Domain decomposition methods in science and engineering XVII*, volume 60 of *Lect. Notes Comput. Sci. Eng.*, pages 247–254. Springer, Berlin, 2008.

[6] C. R. Dohrmann and O. B. Widlund. On the design of small coarse spaces for domain decomposition algorithms. *SIAM J. Sci. Comput.*, 39(4):A1466–A1488, 2017.

[7] A. Heinlein, A. Klawonn, S. Rajamanickam, and O. Rheinbach. *FROSch: A Fast And Robust Overlapping Schwarz Domain Decomposition Preconditioner Based on Xpetra in Trilinos*, pages 176–184. Springer, 2020. Preprint `https://kups.ub.uni-koeln.de/9018/`.

[8] A. Heinlein, A. Klawonn, O. Rheinbach, and F. Röver. A three-level extension for fast and robust overlapping schwarz (FROSch) preconditioners with reduced dimensional coarse space. In preparation.

[9] A. Heinlein, A. Klawonn, O. Rheinbach, and F. Röver. A three-level extension of the GDSW overlapping Schwarz preconditioner in two dimensions. In T. Apel, U. Langer, A. Meyer, and O. Steinbach, editors, *Advanced Finite Element Methods with Applications: Selected Papers from the 30th Chemnitz Finite Element Symposium 2017*, pages 187–204. Springer, Cham, 2019.

[10] A. Heinlein, A. Klawonn, O. Rheinbach, and F. Röver. A three-level extension of the GDSW overlapping Schwarz preconditioner in three dimensions. In R. Haynes, S. MacLachlan, X. Cai, L. Halpern, H. H. Kim, A. Klawonn, and O. Widlund, editors, *Domain Decomposition Methods in Science and Engineering XXV*, pages 185–192, Cham, 2020. Springer International Publishing.

[11] A. Heinlein, A. Klawonn, O. Rheinbach, and O. B. Widlund. Improving the parallel performance of overlapping Schwarz methods by using a smaller energy minimizing coarse space. In P. E. Bjørstad, S. C. Brenner, L. Halpern, H. H. Kim, R. Kornhuber, T. Rahman, and O. B. Widlund, editors, *Domain Decomposition Methods in Science and Engineering XXIV*, pages 383–392, Cham, 2018. Springer.

[12] Pierre Jolivet, Frédéric Hecht, Frédéric Nataf, and Christophe Prud'homme. Scalable domain decomposition preconditioners for heterogeneous elliptic problems. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '13, pages 80:1–80:11, New York, NY, USA, 2013. ACM.

[13] G. Karypis and K. Schloegel. ParMETIS-parallel graph partitioning and sparse matrix ordering library version 4.0. Technical report, University of Minnesota, Department of Computer Science and Engineering Minneapolis, March 2013.

[14] J. Mandel, B. Sousedík, and C. R. Dohrmann. Multispace and multilevel BDDC. *Computing*, 83(2-3):55–85, 2008.

[15] J. Šístek, B. Sousedík, J. Mandel, and P. Burda. Parallel implementation of multilevel BDDC. In *Numerical mathematics and advanced applications 2011*, 2011. 9th European conference on numerical mathematics and advanced applications, Leicester, UK, September 59, 2011.

[16] X. Tu. Three-level BDDC in two dimensions. *International Journal for Numerical Methods in Engineering*, 69(1):33–59, 2007.

[17] The Zoltan2 Project Team. *The Zoltan2 Project Website*.