Optimization and Coarse-Grid Selection for Algebraic Multigrid

by

© **Tareq Uz Zaman**

A thesis submitted to the School of Graduate Studies
in partial fulfilment of the requirements for
the degree of Doctor of Philosophy

Scientific Computing Program
Memorial University of Newfoundland

March 2023

St. John's Newfoundland

Abstract

Multigrid methods are often the most efficient approaches for solving the very large linear systems that arise from discretized PDEs and other problems. Algebraic multigrid (AMG) methods are used when the discretization lacks the structure needed to enable more efficient geometric multigrid techniques. AMG methods rely in part on heuristic graph algorithms to achieve their performance. Reduction-based AMG (AMGr) algorithms attempt to formalize these heuristics.

The main focus of this thesis is to develop effective algebraic multigrid methods. A key step in all AMG approaches is the choice of the coarse/fine partitioning, aiming to balance the convergence of the iteration with its cost. In past work (MacLachlan and Saad, A greedy strategy for coarse-grid selection, SISC 2007), a constrained combinatorial optimization problem was used to define the "best" coarse grid within the setting of two-level reduction-based AMG and was shown to be NP-complete. In the first part of the thesis, a new coarsening algorithm based on simulated annealing has been developed to solve this problem. The new coarsening algorithm gives better results than the greedy algorithm developed previously.

The goal of the second part of the thesis is to improve the classical AMGr method. Convergence factor bounds do not hold when AMGr algorithms are applied to matrices that are not diagonally dominant. In this part of our research, we present modifications to the classical AMGr algorithm that improve its performance on such matrices. For non-diagonally dominant matrices, we find that strength of connection plays a vital role in the performance of AMGr. To generalize the diagonal approximations of A_{FF} used in classical AMGr, we use a sparse approximate inverse

(SPAI) method, with nonzero pattern determined by strong connections, to define the AMGr-style interpolation operator, coupled with rescaling based on relaxed vectors. We present numerical results demonstrating the robustness of this approach for non-diagonally dominant systems.

In the third part of this research, we have developed an improved deterministic coarsening algorithm that generalizes an existing technique known as Lloyd's algorithm. The improved algorithm provides better control of the number of clusters than classical approaches and attempts to provide more "compact" groupings.

General Summary

Numerical methods are essential to approximate the solutions of mathematical models that describe real-world phenomena. The main goal of my thesis is to develop numerical methods. The different parts of the thesis are about understanding and solving different problems essential for the improvement of certain numerical methods.

Imagine a network of people where a person is connected to at most four other persons. An interesting question to ask is how many people must be removed from the network to ensure that each person left in the network loses at least one connection? The question can be generalized asking to find the minimum number of people that should be removed from a network, where each person is connected to at most χ people, so that each person present in the network loses at least κ connections. Though there is nothing to do with people in my thesis, the first part of the thesis solves a similar combinatorial optimization problem in the process of improving a numerical method.

The second part of this research generalizes an existing numerical method and extends its applicability to a wider class of problems. Here we try to understand the reasons that impede a particular numerical method to approximate better solutions for many different problems and improve the method to make it applicable to a larger class of problems.

The third part of the thesis is about clustering or aggregation of nodes in a network graph. If the nodes in the graph represent people, it would be a network of people. Our goal is to make the clusters better rounded, well-centered, and uniform. We improve an existing algorithm to form balanced clusters of nodes in a network graph.

The new algorithm provides control over the number of clusters and leads to uniform and well-centered partitions of the graph.

To

Nadira Easmin Nina
Arshad Uz Zaman
Nazmun N Sohana
Areeba Rameen
Arfa Raida
Mahid Aariz
Wasi
Hasan
Tanvir
Scott M.
Enamul H.

- life could take a different path
without any one of you.

Acknowledgements

First and foremost, I would like to thank my supervisor Dr. Scott MacLachlan. He fulfilled my craving to formally change the direction of my area of study to the intersection of mathematics and computer science by giving me the opportunity to work on my PhD. He also nurtured me to grow up in this field. He taught me and guided me to learn and understand the concepts necessary for my research works. His guidance, help, support, and ideas were indispensable to complete my research works. His contribution to my PhD is beyond expression. Also, Scott was a constant source of my motivation and encouragement to continue my research works. He took a lot of patience to provide detailed and thoughtful answers to all of my questions regardless of whether the questions were vague, obvious, or interesting.

I would like to thank my co-advisors, Dr. Luke Olson and and Dr. Matthew West from University of Illinois Urbana-Champaign (UIUC) for giving me the opportunity to work with them. Their discussions on different research topics in the group meetings expanded my knowledge in numerical methods and machine learning. Their suggestions, ideas, help, and support were crucial to this research. They also helped to greatly improve the quality of graphs, and publications.

Scott, Luke, and Matt are amazing supervisors. Their deep knowledge in their own fields always stirs me up to study and learn. Their ethics and compassion inspires me to be a better human. I learned from them how to teach and supervise students with love and compassion. I will try my best to do the same to my students in future.

I want to thank my MEng supervisor Dr. M. Enamul Hossain, who was the reason for my coming to Memorial University. He provided me the opportunity to start working with Dr. Scott. I would also like to thank my other committee members, Dr. George Miminis, Dr. Ron Haynes, and Dr. Alex Bihlo. I learned a lot from their amazing class lectures that were helpful to make me confident to start my research works. I would like to thank Nancy Bishop, who was our interdisciplinary programs assistant. She always wanted to do something to help and support to the students in her department. They are great people and my well-wishers.

I am also thankful to my virtual labmates Nicolas Nytko and Ali Taghibakhshi from UIUC. Their discussions in the meetings, and their ideas and ways to solve their re-

search problems enriched my knowledge.

I would like to thank my mother Nadira Easmin Nina and my father Arshad Uz Zaman for their constant support during my PhD. My parents have a dream to see me as one of the greatest scientists or philosophers in the world and they have been praying at least five times each day for that since I was born. Though my only capability is to carry their dream for my children and to hope their dream to be fulfilled one day by one of our descendents, their dream was a great source of my positive energy and motivation during my PhD. I am thankful to my younger brother Maruf Hosain Tanvir for taking care of my parents in my absence. Video conversation with them is always a source of joy and happiness for my children while we are living abroad. I would also like to thank my mother-in-law, Mahfuza Begum, for her regular communication and inspiration. I want to thank my elder brother Wasi Uz Zaman who is proud of my higher education. We grew up together like twins. He never hesitated to sacrifice the better things for me when we were living together. I had another younger brother, Hasan Uz Zaman, whose age neither went nor will ever go past two. I cannot forget his unbearable and enormous sufferings in his last days. After my mother, I was his most favourite one and my name was one of the very few words that he learned to say first. He called me again and again in his last days probably hoping that I could remove his sufferings. No, no one could and I was only nine then. My brother's and my mother's acute anguish at that time is a source of my strength in my hard days.

I am incredibly thankful to my beautiful wife, Nazmun Nahar Sohana, for her immense sacrifice to successfully complete my PhD. Her emotional support, love, patience, encouragement, and understanding during the dark times cannot be expressed in words. She sacrificed her career to take care of our children during my masters and PhD. She is the person who had to sacrifice the most for my study and research during my masters and PhD. Sohana, my pagli, you earned this degree right along with me. I am deeply thankful to my amazing twin daughters, Areeba Rameen, Arfa Raida, and my son Mahid Aariz. They have been a continuous source of my joy and happiness throughout my PhD. It was difficult time for Areeba and Arfa to live without their grandparents, uncles, aunts, cousins, and old friends. They had to pass even more difficult time when Mahid was born. As a newborn child, Mahid needed much of our time and concentration, leaving very little of those for Areeba and Arfa. My daughters could understand my emotional ups and downs, and frustrations. They are my

very close friends whom I can share most of my thoughts and feelings. Mahid was also a great source of my emotional support. Sometimes when I became extremely frustrated, when life seemed meaningless, and I wanted to hide my feelings from my wife and daughters, I would hug my son tightly standing by the window looking at the far horizon and it would remove all of my frustrations. I am eternally grateful to my wife and children for their sacrifice, patience, and understanding. Their life could be more enjoyable than it was during my PhD due to my PhD.

Statement of Contributions

The work represented in Chapter 3 is the result of collaborative research between Tareq Zaman, Scott MacLachlan, Luke Olson, and Matthew West, with its intellectual property equally co-owned by all. Tareq Zaman is the primary author of all of this work, followed by faculty coauthors (alphabetically).

The work represented in Chapters 4 and 5 is the result of collaborative research between Tareq Zaman, Nicolas Nytko, Ali Taghibakhshi, Scott MacLachlan, Luke Olson, and Matthew West, with its intellectual property equally co-owned by all. Tareq Zaman is the primary author of all of this work, with student coauthors listed next (alphabetically), followed by faculty coauthors (also alphabetically).

Contents

Abstract ii

General Summary iv

Acknowledgements vii

Statement of Contributions x

List of Tables xiv

List of Figures xix

1 Introduction 1

1.1 Mathematical and discretized models 1
1.2 Direct methods for solving linear systems 4
1.3 Iterative methods and preconditioning for solving linear systems . . . 7
1.4 Literature review 11
1.4.1 Classical coarsening 11
1.4.2 Aggregation-based coarsening 12
1.4.3 AMGr 13
1.5 Contributions of this Thesis 14
1.6 Outline 16

2 Background 22

2.1 (Geometric) multigrid 22
2.2 Algebraic multigrid 25
2.3 Smoothed aggregation AMG 28
2.3.1 Greedy aggregation 29

2.3.2 Maximal independent set based aggregation 30
2.3.3 Standard Lloyd aggregation 32
2.4 Classical AMGr 33
2.4.1 Coarsening in AMGr 36

3 Coarse-Grid Selection Using Simulated Annealing 41
3.1 Introduction 42
3.2 AMG coarsening 45
3.2.1 Classical coarsening 45
3.2.2 Greedy coarsening and underlying optimization 46
3.3 Simulated annealing 51
3.3.1 Idea and generic algorithm 52
3.3.2 Adaptation to coarse/fine partitioning 54
3.3.3 Localization and Gauss-Seidel variants 57
3.3.4 Benchmark results 60
3.4 Results 62
3.4.1 Structured-grid discretizations with geometrically structured subdomains 62
3.4.2 Structured-grid discretizations with algebraically chosen subdomains 71
3.4.3 Anisotropic problems on structured grids 75
3.4.4 Discretizations on unstructured grids 77
3.4.5 Convection-diffusion problems on structured grids 81
3.5 Conclusions and future work 82

4 Generalizing Reduction-Based Algebraic Multigrid 92
4.1 Introduction 93
4.2 Algebraic multigrid 96
4.2.1 Classical (Ruge-Stüben) AMG 97
4.2.2 Reduction-based algebraic multigrid (AMGr) 98
4.2.2.1 Coarsening in AMGr 101
4.2.3 Failure of AMGr for anisotropic diffusion 102
4.3 Sparse Approximate Inverse (SPAI) methods 104
4.4 Generalizing AMGr 106
4.4.1 Algebraic Coarsening 110
4.4.2 The Generalized AMGr algorithm 114

4.5 Results . . . 117
4.5.1 Isotropic Poisson problem . . . 118
4.5.2 Four-quadrant problems . . . 121
4.5.3 Unstructured anisotropic diffusion . . . 123
4.5.4 Multilevel Results . . . 124
4.6 Conclusions and future work . . . 128

5 Generalizing Lloyd's Algorithm for Graph Clustering 135
5.1 Introduction . . . 136
5.2 Clustering in algebraic multigrid . . . 137
5.2.1 Greedy clustering . . . 138
5.2.2 Maximal independent set based clustering . . . 140
5.2.3 Standard Lloyd clustering . . . 141
5.2.3.1 Theoretical observations . . . 142
5.3 Balanced Lloyd clustering . . . 143
5.3.1 Balanced algorithms . . . 145
5.3.2 Rebalancing clustering . . . 147
5.3.3 Theoretical observations . . . 151
5.4 Numerical Results . . . 155
5.4.1 Varying cluster numbers . . . 157
5.4.2 Tiebreaking . . . 157
5.4.3 Rebalancing . . . 159
5.4.4 Algebraic multigrid convergence . . . 160
5.4.5 Additional problems in Algebraic Multigrid . . . 163
5.5 Conclusions and extensions . . . 164

6 Conclusions and Future Work 172

List of Tables

1.1 Physical meaning of the terms in Equation (1.1) 2

3.1 Performance of two-level AMGr on test matrices from discretizations of the 2D Laplacian. 71
3.2 Performance of three-level and multilevel AMGr on test matrices from discretizations of the 2D Laplacian. 73
3.3 Performance of multilevel AMGr on test matrices from discretizations of the 2D Laplacian using 200 and 2000 SA steps per DoF. 74
3.4 Performance of multilevel AMGr on test matrices from discretizations of the 2D Laplacian with jumping coefficients. 74
3.5 Performance of two-level AMGr for the anisotropic diffusion problem with $\delta = 10^{-6}$ and $\theta = \pi/3$ on structured meshes. 78
3.6 Performance of three-level AMGr for the anisotropic diffusion problem with $\delta = 10^{-6}$ and $\theta = \pi/3$ on structured meshes. 78
3.7 Performance of two- and three-level AMGr for isotropic problem on unstructured meshes. 80
3.8 Performance of two-level and three-level AMGr for anisotropic problem on unstructured meshes. 81
3.9 Performance of two-level AMGr for convection-diffusion problems on structured meshes. 83
3.10 Number of SA steps per DoF per sweep used for numerical results in left-hand panel of Figure 3.3. 89
3.11 Number of SA steps per DoF per sweep used for numerical results in right-hand panel of Figure 3.3. 89
3.12 Number of SA steps and SA steps per DoF per sweep used for numerical results in left-hand panel of Figure 3.7. 90

3.13 Number of SA steps and SA steps per DoF per sweep used for numerical results in right-hand panel of Figure 3.7. 90
3.14 Number of SA steps per DoF per sweep used for numerical results in right-hand panel of Figure 3.8. 90
3.15 Number of SA steps and SA steps per DoF per sweep used for numerical results in Figure 3.10. 91
3.16 Number of SA steps per DoF and SA steps per DoF per sweep and subdomain sizes used for two-level numerical results in Table 3.5. . . 91
3.17 Number of SA steps per DoF and SA steps per DoF per sweep and subdomain sizes used for two-level numerical results in Tables 3.7 and 3.8. 91

4.1 Performance of two-level AMGr for the anisotropic diffusion problem. 103
4.2 Two-level AMGr convergence factors for anisotropic FE discretization using semi-coarsening in the y direction by a factor of three. Interpolation of F-nodes uses the SPAI approximation to $A_{FF}^{-1}A_{FC}$ and the SPAI approximation to A_{FF}^{-1} is used for relaxation. 107
4.3 Two-level AMGr convergence factors for anisotropic FE discretization using semi-coarsening in the y direction by a factor of three. Interpolation of F-nodes is based on a SPAI approximation to $\hat{A}_{FF}^{-1}\hat{A}_{FC}$ and relaxation uses a SPAI approximation to $\hat{A}_{FF}^{-1}$, **where $\hat{A}$ is the "lumped" matrix of strong connections computed from** A. . . 108
4.4 Two-level AMGr convergence factors for anisotropic FE discretization using semi-coarsening in the y direction by a factor of 3. Here, F-nodes are interpolated using the SPAI approximation to $\hat{A}_{FF}^{-1}\hat{A}_{FC}$, **postprocessed to exactly interpolate the constant vector**, and the SPAI approximation to $\hat{A}_{FF}^{-1}$ is used for relaxation, where $\hat{A}$ is the "lumped" matrix of strong connections computed from A. 109
4.5 Two-level AMGr convergence factors for anisotropic FE discretization using semi-coarsening in the y direction by a factor of 3. Here, F-nodes are interpolated using the SPAI approximation to $\hat{A}_{FF}^{-1}\hat{A}_{FC}$, **postprocessed to exactly interpolate a relaxed vector**, and the SPAI approximation to $\hat{A}_{FF}^{-1}$ is used for relaxation, where $\hat{A}$ is the "lumped" matrix of strong connections computed from A. 110

4.6 Two-level AMGr convergence factors for anisotropic FE discretization using semi-coarsening in the y direction by a factor of 3. Here, F-nodes are interpolated using the SPAI approximation to $\hat{A}_{FF}^{-1}\hat{A}_{FC}$, postprocessed to exactly interpolate a relaxed vector, and the SPAI approximations to $\hat{A}_{FF}^{-1}$ and $\hat{A}_{CC}^{-1}$ are used **for FCF-relaxation**, where $\hat{A}$ is the "lumped" matrix of strong connections computed from A. 111

4.7 Two-level AMGr convergence factors and corresponding complexities for anisotropic FE discretization **using simulated annealing coarsening with $\eta = 0.65$.** F-nodes are interpolated using the SPAI approximation to $\hat{A}_{FF}^{-1}\hat{A}_{FC}$, postprocessed to exactly interpolate a relaxed vector, and the SPAI approximations to $\hat{A}_{FF}^{-1}$ and $\hat{A}_{CC}^{-1}$ are used for FCF-relaxation, where $\hat{A}$ is the "lumped" matrix of strong connections computed from A. 112

4.8 Two-level AMGr convergence factors and corresponding complexities for anisotropic FE discretization **using simulated annealing coarsening with $\eta = 0.75$.** F-nodes are interpolated using the SPAI approximation to $\hat{A}_{FF}^{-1}\hat{A}_{FC}$, postprocessed to exactly interpolate a relaxed vector, and the SPAI approximations to $\hat{A}_{FF}^{-1}$ and $\hat{A}_{CC}^{-1}$ are used for FCF-relaxation, where $\hat{A}$ is the "lumped" matrix of strong connections computed from A. 113

4.9 Two-level AMGr convergence factors and corresponding complexities for anisotropic FE discretization using simulated annealing coarsening with $\eta = 0.65$. F-nodes are interpolated using the SPAI approximation to $\hat{A}_{FF}^{-1}\hat{A}_{FC}$, postprocessed to exactly interpolate a relaxed vector, and the SPAI approximations to $\hat{A}_{FF}^{-1}$ and $\hat{A}_{CC}^{-1}$ are used for FCF-relaxation, where $\hat{A}$ is the "lumped" matrix of strong connections computed from A. **Interpolation truncation with $\zeta = 0.2$ is used.** 114

4.10 **Three-level** AMGr convergence factors and corresponding complexities for anisotropic FE discretization using simulated annealing coarsening with $\eta = 0.65$. F-nodes are interpolated using the SPAI approximation to $\hat{A}_{FF}^{-1}\hat{A}_{FC}$, postprocessed to exactly interpolate a relaxed vector, and the SPAI approximations to $\hat{A}_{FF}^{-1}$ and $\hat{A}_{CC}^{-1}$ are used for FCF-relaxation, where $\hat{A}$ is the "lumped" matrix of strong connections computed from A. 115

4.11 **Three-level** AMGr convergence factors and corresponding complexities for anisotropic FE discretization using simulated annealing coarsening with $\eta = 0.65$. F-nodes are interpolated using the SPAI approximation to $\hat{A}_{FF}^{-1}\hat{A}_{FC}$, postprocessed to exactly interpolate a relaxed vector, and the SPAI approximations to $\hat{A}_{FF}^{-1}$ and $\hat{A}_{CC}^{-1}$ are used for FCF-relaxation, where $\hat{A}$ is the "lumped" matrix of strong connections computed from A. **Interpolation truncation with $\zeta = 0.2$ is employed.** 115
4.12 AMGr convergence factors and complexities for **isotropic FE discretization on structured grids**. Results for classical (diagonal D_{FF}) AMGr appear in the first block column, followed by those for the two-level and three-level SPAI-based algorithm in Section 4.4.2 in following block columns, using $\zeta = 0$. 120
4.13 AMGr convergence factors and complexities for isotropic FE discretization on structured grids, using the two-level and three-level SPAI-based algorithm in Section 4.4.2, **with** $\zeta = 0.25$. 120
4.14 AMGr convergence factors and complexities for isotropic FE discretization on **unstructured grids**, using the two-level and three-level SPAI-based algorithm in Section 4.4.2. Results in the left-most block column show two-level results with no interpolation truncation ($\zeta = 0$), while the other block columns show results with $\zeta = 0.25$. 121
4.15 **Two-level** AMGr convergence factors and complexities for the **four-quadrant problem** using the SPAI-based algorithm in Section 4.4.2, with $\zeta = 0.25$. 123
4.16 **Three-level** AMGr convergence factors and complexities for the **four-quadrant problem** using the SPAI-based algorithm in Section 4.4.2, with $\zeta = 0.25$. 123
4.17 Two- and three-level AMGr convergence factors and complexities for the **anisotropic diffusion problem on unstructured meshes** using the SPAI-based algorithm in Section 4.4.2, with $\zeta = 0.25$. 125

4.18 **Multi-level** AMGr convergence factors and corresponding complexities for anisotropic FE discretization using **greedy coarsening** with $\eta = 0.65$. F-nodes are interpolated using the SPAI approximation to $\hat{A}_{FF}^{-1}\hat{A}_{FC}$, postprocessed to exactly interpolate a relaxed vector, and the SPAI approximations to $\hat{A}_{FF}^{-1}$ and $\hat{A}_{CC}^{-1}$ are used for FCF-relaxation, where $\hat{A}$ is the "lumped" matrix of strong connections computed from A. Interpolation truncation with $\zeta = 0.2$ is employed. **Estimates of the eigenvalues are used in relaxation.** 126

4.19 **Multi-level** AMGr convergence factors and complexities for the **four-quadrant problem** using the SPAI-based algorithm in Section 4.4.2, with $\zeta = 0.25$. **Greedy coarsening and the heuristic eigenvalue estimates are used.** 127

4.20 **Multi-level** AMGr convergence factors and complexities for **anisotropic diffusion problem on unstructured meshes** using the SPAI-based algorithm in Section 4.4.2, with $\zeta = 0.25$. **Greedy coarsening, for three values of η, and the heuristic eigenvalue estimates are used.** 127

5.1 Additional examples. *ACTII mesh credit*: Mike Anderson at UIUC. . 165

5.2 List of symbols. Here $\mathcal{P}()$ denotes the power set and $\overline{\mathbb{R}}$ is the extended reals. 171

List of Figures

1.1 Change in $\log_{10} ||e||_\infty$ with the number of iterations for the Gauss-Seidel iteration applied to Eq. (1.5) on a 32×32 mesh for 100 iterations. $\boldsymbol{u}^{(1,1)}$, $\boldsymbol{u}^{(5,5)}$, $\boldsymbol{u}^{(10,10)}$, and $\boldsymbol{u}^{(28,28)}$ are taken as the initial guesses. . . . 10

1.2 Change in $\log ||e||_\infty$ with the number of iterations for the Gauss-Seidel iteration applied to Eq. (1.5) on 32×32, 64×64, and 128×128 mesh and an initial guess $\frac{1}{4}(\boldsymbol{u}^{(1,1)} + \boldsymbol{u}^{(5,5)} + \boldsymbol{u}^{(10,10)} + \boldsymbol{u}^{(28,28)})$ for 100 iterations. 10

1.3 Splitting of C- and F- points for uniform 32×32 meshes from classical Ruge-Stüben (RS) coarsening. At left, coarsening for the FE discretization of isotropic-diffusion. At right, coarsening for the FE discretization of anisotropic-diffusion with strength of anisotropy 10^{-6} and direction of anisotropy $\pi/3$. Fine-grid DoFs are denoted by filled grey dots; those that are in C are marked with black circles. 12

1.4 Example of greedy aggregation for uniform 32×32 meshes. At left, aggregation for the FE discretization of isotropic-diffusion. At right, aggregation for the FE discretization of anisotropic-diffusion with strength of anisotropy 10^{-6} and direction of anisotropy $\pi/3$. 13

2.1 Change in $\log ||e||_\infty$ with the number of iterations for the Gauss-Seidel iteration and two-level multigrid applied to Eq. (1.5) on 32×32, 64×64, and 128×128 mesh and an initial guess $\frac{1}{4}(\boldsymbol{u^{(1,1)}} + \boldsymbol{u^{(5,5)}} + \boldsymbol{u^{(10,10)}} + \boldsymbol{u^{(28,28)}})$ for 100 iterations. 24

3.1 Splitting of C- and F- points for 11×11 meshes from optimization "by hand". At left, X-pentomino coarsening for five-point FD scheme. At right, 3×3 brick coarsening for nine-point FE scheme. At left, we color by the X-pentominos and at right, we color by the 3×3 bricks. C-points are represented by the black circles. 61

3.2 Maximum value of $|F|/|\Omega|$ with number of annealing steps per DoF per GS sweep for different numbers of annealing steps per DoF for the 32×32 uniform-grid five-point finite-difference discretization. Each panel shows a different size of geometrically chosen subdomain: 2×2 (top-left), 3×3 (top-right), 4×4 (bottom-left), 6×6 (bottom-right). 63
3.3 Left: Change in $|F|/|\Omega|$ with subdomain size for the 32×32 uniform-grid five-point finite-difference discretization, using geometric subdomains, with 5000, 200 000, and 2 000 000 total SA steps per DoF. Right: Change in maximum number of F-points with number of total SA steps per DoF for this problem using 6×6 subdomains. 65
3.4 Change in $|F|/|\Omega|$ (left) and temperature (right) with number of annealing steps for the 32×32 uniform-grid finite-difference discretization of the Laplacian. At left, the case of SA with five SA steps per DoF per GS sweep using 6×6 subdomain size is shown, along with greedy as a baseline. Note that the vertical axis on the left plot uses a mixed log-linear scale for clarity, with a break at -10^{-2}. 66
3.5 Change in $|F|/|\Omega|$ with SA steps per DoF for a fixed temperature decay rate (left) and temperature (right) for the 32×32 uniform-grid finite-difference discretization of the Laplacian. Here, one SA step per DoF per sweep is used, and the temperature decay rate is fixed, with $\alpha = 0.1^{(1.0/(200000 \times 32 \times 32))}$. The circles in the right figure show the stopping temperatures for the annotated SA steps per DoF. 66
3.6 Grid partitioning for 32×32 uniform grid with five-point FD stencil using two million SA steps per DoF. At left, the partitioning is generated using primarily 4×4 subdomains, with 25 SA steps per DoF per cycle. At right, the partitioning is generated using 6×6 subdomains, with 5 SA steps per DoF per cycle. The grid at left has 814 F-points, while that at right has 816. 67
3.7 Change in $|F|/|\Omega|$ with mesh size for FD (left) and FE (right) discretizations. 68

3.8 Quality of coarsening for the 32×32 uniform-grid nine-point finite-element discretization, using geometric subdomains. Left: Maximum value of $|F|/|\Omega|$ with number of annealing steps per DoF per GS sweep for different numbers of annealing steps per DoF using 5×5 geometric subdomains. Right: Change in $|F|/|\Omega|$ with subdomain size and total number of SA steps per DoF. 69
3.9 Grid partitioning for 32×32 uniform grid with nine-point FE stencil using two million SA steps per DoF. At left, the partitioning is generated using 5×5 subdomains, with one SA step per DoF per cycle. At right, the partitioning is generated using 6×6 subdomains, also with one SA step per DoF per cycle. The grid at left has 814 F-points, while that at right has 811. 70
3.10 Change in maximum number of F-points with subdomain size for algebraic subdomain selection on 32×32 meshes. Left and right figures are for FD and FE schemes, respectively, showing the largest value of $|F|/|\Omega|$ attained over experiments with fixed subdomain size and varying the total number of SA steps per DoF and SA steps per sweep, as in the geometric subdomain case. 72
3.11 Partitioning for the isotropic problem on a 31×31 uniform grid with jumping coefficients, generated using 200 000 SA steps per DoF, 1 SA step per DoF per GS sweep, and subdomains with an average of 36 points per subdomain. C-points are represented by the black circles. . 75
3.12 Grid partitioning for an anisotropic diffusion problem with $\delta = 10^{-6}$ and $\theta = \pi/3$, discretized on a 32×32 mesh using the bilinear FE stencil. The initial partitioning, at left, is generated using algebraic subdomains averaging 20 points per subdomain, using 2 000 000 SA steps per DoF and 1 SA step per DoF per GS sweep, and has 664 F-points and 360 C-points. At right is the grid augmented using the second pass of the classical AMG algorithm, resulting in 343 F-points and 681 C-points. 77
3.13 Partitioning for the isotropic problem on the unstructured mesh with 1433 points, generated using 1 000 000 SA steps per DoF, 5 SA steps per DoF per GS sweep, and subdomains with an average of 20 points per subdomain. 79

3.14 Partitioning for the anisotropic problem on an unstructured mesh containing 798 points. The partitioning at left was generated using 500 000 SA steps per DoF, with 1 SA step per DoF per GS sweep, on subdomains with an average size of 36 points per subdomain, yielding 524 F-points and 274 C-points. At right, this partitioning is augmented by the second pass of classical AMG coarsening, resulting in 263 F-points and 535 C-points. 81
3.15 Partitioning for convection-diffusion problems on uniform grids, for the grid-aligned case, at left, and the non-grid-aligned case, at right. In both cases, we depict the partitioning for $\varepsilon = 10^{-5}$ and $N = 32$, generated using 200 000 SA steps per DoF. 83

4.1 Algebraic coarse-fine partitionings for the FE discretization of anisotropic-diffusion with $\theta = 0$ on uniform 32×32 grid. At left, partitioning with $\eta = 0.65$. At right, partitioning with $\eta = 0.75$. Fine-grid DoFs (points in Ω) are denoted by filled grey dots; those that are in C are marked with black circles. 113
4.2 Trade-off between two-level convergence factor and operator complexities as a function of ζ, for isotropic Poisson on a uniform 32×32 grid (at left) and on the unstructured triangulation with 1433 DoFs (at right). 120
4.3 Visualization of the strong connections for the four-quadrant problems; Problem 1 (left), Problem 2 (middle), and Problem 3 (right). 122
4.4 Unstructured triangulation containing 798 points from Brannick and Falgout [19], and its partitioning using $\eta = 0.65$. Fine-grid DoFs (points in Ω) are denoted by filled grey dots; those that are in C (282 points) are marked with black circles. 124

5.1 Example clusterings. 136
5.2 Two example clusterings from Lloyd clustering on a 6×6 mesh. . . . 144
5.3 Algorithm dependence. Rebalancing components are highlighted in red. 146
5.4 Work per digit (WPD) of accuracy and convergence ρ for clustering sizes ranging from 3–19 points per cluster (on average) using rebalanced Lloyd clustering. The average over 100 runs is marked $\circ$ and a trendline from a smoothed cubic spline is given for the mean (solid). 158
5.5 Example clustering patterns with the number of clusters ranging from 5 to 250 using rebalanced Lloyd clustering. 158

5.6 Distribution of the number of clusters having zero diameter for balanced Lloyd clustering with or without tiebreaking for a 64×64 mesh. 159
5.7 Distribution of the standard deviation in the number of nodes and distribution of energy for balanced Lloyd clustering with or without tiebreaking on a 64×64 mesh. 160
5.8 Distribution of standard deviation in diameters, distribution of standard deviation in number of nodes, and distribution in energy for different clustering methods for a 64×64 mesh. 161
5.9 (Left) Difference between maximum and minimum diameters of clusters averaged over 1000 samples; (Right) Energy per node averaged over 1000 samples. The shaded regions mark one standard deviation from the mean. 161
5.10 Localizing β to each cluster via (5.11). 163
5.11 Example convergence for two-level AMG with different clusterings. . . 163
5.12 Example clustering and restriction matrix. 170

Chapter 1

Introduction

Scientific simulation has become indispensable for studying many real or hypothetical systems. In this process, the physical or hypothetical system is mimicked by a mathematical model and it is hoped that this model will closely approximate the dynamic behaviour of the actual system. Computer simulation is important as there are many complicated physical processes that are very difficult to study directly and also difficult to simulate in laboratories. Starting from meteorology and nuclear physics, computer simulations are now being used in a vast array of disciplines and can be expected to be used in almost every scientific and engineering discipline in future.

1.1 Mathematical and discretized models

Generally, a physical or hypothetical system is first represented by a set of mathematical equations called the mathematical model of the system. Mathematical models are useful tools to understand and study the properties of the original system. Very often these mathematical models consist of systems of differential equations (DEs), especially partial differential equations (PDEs).

As an example, a very common family of differential equations in mathematical models take the form

$$-\nabla \cdot \boldsymbol{K}(x,y)\nabla V(x,y) = \alpha \frac{\partial V(x,y)}{\partial t}, \tag{1.1}$$

which can be used to model heat flow, fluid flow in porous media, or displacement in a hanging bar, with α and $\boldsymbol{K}$ representing physical parameters. Here, $(x,y) \in \mathbb{R} \times \mathbb{R}$ represents coordinates, and $V(x,y) : \mathbb{R} \times \mathbb{R} \longmapsto \mathbb{R}$ is a function of (x,y).

Also, $\boldsymbol{K}(x,y) = QHQ^T$ with $Q = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$ being a plane rotation and $H = \begin{bmatrix} \delta & 0 \\ 0 & 1 \end{bmatrix}$. Furthermore, θ and δ represent direction and strength of anisotropy, respectively.

The physical meanings of the different terms of Equation (1.1) is tabulated in Table 1.1 for three different physical systems that can be modeled by this equation. The mathematical model is derived using conservation principles and equations of state. The equation beautifully expresses the relationship between the temperature gradient and temperature change with time, or the pressure gradient and pressure change with time, or the change in time of strain and displacement. In principle, we can use this equation to compute the distribution of temperature, pressure, or displacement at different points in space and time.

Table 1.1: Physical meaning of the terms in Equation (1.1)

Terms	**Physical Meaning**		
	Heat flow	**Fluid flow**	**Hanging bar**
V	TEMPERATURE	PRESSURE	DISPLACEMENT
∇V	TEMPERATURE GRADIENT	PRESSURE GRADIENT	STRAIN
$\boldsymbol{K}\nabla V$	HEAT FLUX	VOLUMETRIC FLUX OR DARCY VELOCITY	STRESS
$-\nabla \cdot \boldsymbol{K}\nabla V$	$\frac{\text{HEAT ACCUMULATION}}{\text{VOLUME/TIME}}$	$\frac{\text{VOLUME ACCUMULATION}}{\text{VOLUME/TIME}}$	$\frac{\text{FORCE}}{\text{VOLUME}}$
$\frac{\partial V}{\partial t}$	$\frac{\text{TEMPERATURE CHANGE}}{\text{TIME}}$	$\frac{\text{PRESSURE CHANGE}}{\text{TIME}}$	$\frac{\text{DISPLACEMENT CHANGE}}{\text{TIME}}$

In many cases these mathematical models cannot be solved analytically. Hence, the PDEs are transformed into a related algebraic system of equations using some discretization method. Spatial discretization, required for the left hand side of Equation (1.1), can be made using the finite difference (FD), finite element (FE), or finite volume (FV) methods. In a finite difference discretization, Taylor's theorem is used to transform a PDE into an algebraic system of equations, by approximating the derivatives of a function at a set of points in the function domain.

To explain better, we will work through an example for the two-dimensional Pois-

son equation

$$\begin{aligned} -V_{xx} - V_{yy} &= 2(x - x^2 + y - y^2), \quad 0 < x, y < 1, \\ V(x,0) = V(x,1) = V(0,y) = V(1,y) &= 0, \quad 0 \le x \le 1, 0 \le y \le 1. \end{aligned} \tag{1.2}$$

This equation can, of course, be solved analytically. However, we use this as a motivating example for numerical solution techniques. Among the finite-difference (FD), finite-element (FE) and finite-volume (FV) methods that can be employed to numerically solve the equation, we consider the finite-difference method for its simplicity. The domain of the problem $\{(x,y) : 0 \le x \le 1, 0 \le y \le 1\}$ is partitioned into p subintervals in the x-direction and q subintervals in the y-direction. The grid points here are introduced as $(x_i, y_j) = (ih_x, jh_y)$, where $h_x = \frac{1}{p}$ and $h_y = \frac{1}{q}$. The grid points are the points where we intend to compute approximations to the values of V. Hence, the grid can also be considered as a set of unknowns. We denote this two-dimensional grid by Ω, and note that it has a total of n interior grid points, where $n = (p-1)(q-1)$. The grid Ω here is our "fine grid" and the grid points are fine-grid points. Replacing the differential equation (1.2) at each of the n interior grid points by a second-order finite difference approximation gives the following system of linear equations

$$\begin{aligned} \frac{-v_{i-1,j} + 2v_{i,j} - v_{i+1,j}}{h_x^2} + \frac{-v_{i,j-1} + 2v_{i,j} - v_{i,j+1}}{h_y^2} &= 2(x_i - x_i^2 + y_j - y_j^2), \\ v_{i,0} = v_{i,q} = v_{0,j} = v_{p,j} &= 0, \quad 1 \le i \le p-1, 1 \le j \le q-1. \end{aligned} \tag{1.3}$$

where the vector $\boldsymbol{v}$ represents the approximate solution of Eq. (1.2). Here, $v_{i,j}$ is an approximation to the exact solution $V(x_i, y_j)$. However, $\boldsymbol{v}$ is the exact solution of the system of linear equations represented by Eq. (1.3). Considering the lexicographic ordering of the $n = (p-1)(q-1)$ unknowns by lines of constant i and collecting the unknowns of the i-th column of the grid in the vector $\boldsymbol{v_i} = (v_{i,1}, ..., v_{i,q-1})^T$ for $1 \le i \le p-1$, the system of equations (1.3) can be written in block matrix form as

$$\begin{bmatrix} Y & -aI & & \\ -aI & Y & -aI & \\ \ddots & \ddots & \ddots & \\ & & & -aI \\ & & -aI & Y \end{bmatrix} \begin{bmatrix} \boldsymbol{v_1} \\ \boldsymbol{v_2} \\ \vdots \\ \boldsymbol{v_{p-2}} \\ \boldsymbol{v_{p-1}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f_1} \\ \boldsymbol{f_2} \\ \vdots \\ \boldsymbol{f_{p-2}} \\ \boldsymbol{f_{p-1}} \end{bmatrix}, \tag{1.4}$$

where

$$Y = \begin{bmatrix} \frac{2}{h_x^2} + \frac{2}{h_y^2} & -\frac{1}{h_y^2} & & & \\ -\frac{1}{h_y^2} & \frac{2}{h_x^2} + \frac{2}{h_y^2} & -\frac{1}{h_y^2} & & \\ \ddots & \ddots & \ddots & & \\ & & & -\frac{1}{h_y^2} & \\ & & -\frac{1}{h_y^2} & \frac{2}{h_x^2} + \frac{2}{h_y^2} & \end{bmatrix}$$

is a $(q-1) \times (q-1)$ tridiagonal matrix, $a = \frac{1}{h_x^2}$, and I is a $(q-1) \times (q-1)$ identity matrix. The matrix in this equation is symmetric, block-tridiagonal, sparse, and has a block dimension $(p-1) \times (p-1)$. Eq. (1.4) can be written as

$$A\boldsymbol{v} = \boldsymbol{f}, \tag{1.5}$$

where $A \in \mathbb{R}^{n \times n}$.

1.2 Direct methods for solving linear systems

Gaussian elimination is a direct method that comes naturally to mind when solving a system of linear equations. The algorithm can solve a system of linear equations of the form $A\boldsymbol{v} = \boldsymbol{f}$ for any nonsingular matrix A. Here, two triangular matrices L and U are computed so that $A = LU$. The factorization of the matrix A into L and U can be done in different ways. Algorithm 1.1 describes the ikj version of the "in-place" LU factorization. In the row reduction process, starting from the second row, each element $A_{i,k}$ prior to the diagonal element in row i is eliminated using a factor that is computed using $A_{i,k}$ and $A_{k,k}$. The element $A_{i,k}$ is overwritten by the factor and each element following $A_{i,k}$ is updated using the factor, and the element in the same column of the k-th row. Hence, the triangular part below the diagonal of the output matrix from Algorithm 1.1 contains the factors used to transform the system $A\boldsymbol{v} = \boldsymbol{f}$ into $U\boldsymbol{v} = \boldsymbol{z}$ where $\boldsymbol{z}$ is the solution of $L\boldsymbol{z} = \boldsymbol{f}$. L is the strictly lower triangular part of the output matrix from Algorithm 1.1 and U is the upper triangular part of this output matrix. Algorithm 1.2 solves the system $L\boldsymbol{z} = \boldsymbol{f}$ and Algorithm 1.3 solves the system $U\boldsymbol{v} = \boldsymbol{z}$. The solution process in Algorithm 1.3 starts solving for v_n from the last equation and subsequently substitutes the known values to compute the unknown values.

Gaussian elimination is a general-purpose algorithm. The downside of the Gaussian elimination is that it is computationally expensive. Algorithm 1.1 shows that

Algorithm 1.1 LU factorization of the matrix A

```
1: function LU-FACTORIZATION(A)
2:     for i ← 2, ..., n do
3:         for k ← 1, ..., i − 1 do
4:             A_{i,k} = A_{i,k} / A_{k,k}
5:             for j ← k + 1, ..., n do
6:                 A_{i,j} = A_{i,j} − A_{i,k} A_{k,j}
7:     return (A)
```

Algorithm 1.2 Solution of $L\mathbf{z} = \mathbf{f}$

```
1: function FORWARD-SUBSTITUTION-SOLVE(L, f)
2:     n ← length of f
3:     z_i ← 0 for all i = 1, ..., n
4:     z_1 ← f_1 / L_{1,1}
5:     for i ← 2, ..., n do
6:         for j ← 1, ..., i − 1 do
7:             f_i = f_i − L_{i,j} z_j
8:         z_i = f_i / L_{i,i}
9:     return z
```

Algorithm 1.3 Solution of $U\mathbf{v} = \mathbf{z}$

```
1: function BACKWARD-SUBSTITUTION-SOLVE(U, z)
2:     n ← length of z
3:     v_i ← 0 for all i = 1, ..., n
4:     v_n ← z_n / U_{n,n}
5:     for i ← n − 1, ..., 1 do
6:         for j ← i + 1, ..., n do
7:             z_i = z_i − U_{i,j} v_j
8:         v_i = z_i / U_{i,i}
9:     return v
```

there are two floating point operations (flops) in the third FOR loop and one flop in the second FOR loop. The total number of floating point operations can be expressed

as

$$\sum_{i=2}^{n}\sum_{k=1}^{i-1}\left(1+\sum_{j=k+1}^{n}2\right),$$

which sums to $\frac{2}{3}n^3 - \frac{1}{2}n^2 - \frac{1}{6}n$. Therefore, the dense implementation of Gaussian elimination requires $O(n^3)$ arithmetic operations to solve a system of equations where n is the number of degrees of freedom (DoFs).

However, the linear systems that come from a discretization process are typically sparse, hence, the sparsity can be exploited to reduce the computational cost. A sparse matrix can be banded meaning that the nonzero entries are confined in a band around the main diagonal. For a sparse matrix, A, if $A_{i,j} = 0$ for $|i-j| > B$, where $B < n$, then the smallest such B is called the bandwidth of A. For a banded sparse matrix, the LU factorization does not need to overwrite the zero elements outside the band in the strictly lower triangular part. Hence, the computation of the factors corresponding to these zero elements and the corresponding updates to the elements in the upper triangular parts are not required. Thus, the zero elements outside the band in upper triangular part need never be updated. Algorithm 1.1 can be modified into Algorithm 1.4 for banded sparse matrices. The total number of flops in the modified algorithm can be computed as

$$\sum_{i=2}^{n}\sum_{k=\max(1,i-B)}^{i-1}\left(1+\sum_{j=k+1}^{\min(k+B,n)}2\right) \leq (2B^2+B)n.$$

Hence, the number of arithmetic operations for the banded factorization is reduced to $O(B^2n)$, which is significantly less than $O(n^3)$ if $B \ll n$.

Algorithm 1.4 LU factorization of the banded sparse matrix A

```
1: function LU-FACTORIZATION-BANDED-SPARSE(A)
2:     for i ← 2, ..., n do
3:         for k ← max(1, i − B), ..., i − 1 do
4:             A_{i,k} = A_{i,k}/A_{k,k}
5:             for j ← k + 1, ..., min(k + B, n) do
6:                 A_{i,j} = A_{i,j} − A_{i,k}A_{k,j}
7:     return (A)
```

The number of arithmetic operations can be further reduced by reordering the rows and columns of the matrix. Such reordering can decrease the bandwidth and

required computational cost. However, Hoffman, Martin, and Rose (1973) proved that the number of arithmetic operations cannot be less than $O(n^{3/2})$ when A comes from discretization on an $\sqrt{n} \times \sqrt{n}$ mesh. This best possible complexity is obtained in the Nested dissection algorithm due to George (1973) [6]. Solvers with greater than linear complexity are impractical for large n.

1.3 Iterative methods and preconditioning for solving linear systems

Classical direct methods work well for smaller systems; however, they are not feasible for large systems due to their high computational cost. Consequently, we introduce iterative methods to reduce the computational complexity. The goal of iterative methods is to solve for n DoFs in $O(n)$ cost. Part of the price that we pay to accomplish this is that the linear system cannot be solved exactly in this cost. However, we do not need to solve the linear system with a higher accuracy than the discretization error that the linear system contains in the approximation of a discretized PDE.

To have a basic concept about iterative methods, consider $\boldsymbol{v}^{(m)}$ to be the m-th guess to the exact solution $\boldsymbol{v}$ of the linear system $A\boldsymbol{v} = \boldsymbol{f}$. If we wanted to have the exact solution in the $(m+1)$-th guess, we would write,

$$\boldsymbol{v}^{(m+1)} = \boldsymbol{v} = \boldsymbol{v}^{(m)} + (\boldsymbol{v} - \boldsymbol{v}^{(m)}) = \boldsymbol{v}^{(m)} + \boldsymbol{e}^{(m)},$$

where $\boldsymbol{e}^{(m)}$ is the error in $\boldsymbol{v}^{(m)}$. At this point, we do not know $\boldsymbol{e}^{(m)}$ as we do not know $\boldsymbol{v}$. However, we can represent $\boldsymbol{e}^{(m)}$ with the residual, $\boldsymbol{r}^{(m)}$, computing $A^{-1}\boldsymbol{r}^{(m)}$ because

$$\boldsymbol{r}^{(m)} = \boldsymbol{f} - A\boldsymbol{v}^{(m)} = A\boldsymbol{v} - A\boldsymbol{v}^{(m)} = A\boldsymbol{e}^{(m)}.$$

To avoid the computation of the inverse of A, as this would lead to direct methods again, the inverse of A can be approximated by some matrix M leading to

$$\boldsymbol{v}^{(m+1)} = \boldsymbol{v}^{(m)} + M(\boldsymbol{f} - A\boldsymbol{v}^{(m)}). \tag{1.6}$$

The relationship between $\boldsymbol{v}^{(m+1)}$ and $\boldsymbol{v}^{(m)}$ can be made more general, writing,

$$\boldsymbol{v}^{(m+1)} = \boldsymbol{v}^{(m)} + \omega_m M(\boldsymbol{f} - A\boldsymbol{v}^{(m)}),$$

where ω_m is a scalar parameter. The approximation, M, can be constructed in a myriad of ways. One approach to construct M is "matrix splitting".

Let's consider that A is split into $A = D - L - U$, where D is a diagonal matrix, L is a strictly lower-triangular matrix and U is a strictly upper triangular matrix. With this splitting, different choices of M lead to different iterative methods. Choosing $M = D$ in Equation (1.6) gives the Jacobi iteration method, and $M = \frac{1}{\omega}D$, where ω is a scalar parameter, leads to the weighted Jacobi iteration. For the Gauss-Seidel (GS) iteration, M in Equation (1.6) is chosen as $D - L$. In successive over-relaxation (SOR), $M = \frac{1}{\omega}D - L$.

Let's go back to our example problem (Equation (1.2)) and solve Equation (1.5) using Gauss-Seidel iterations to show the performance of standard iterative methods. It can be shown that the eigenvectors of A can be written as $\boldsymbol{u}^{(k,l)} = \sin(k\pi x)\sin(l\pi y)$. $\boldsymbol{u}^{(k,l)}$ can be written in vector form as

$$\boldsymbol{u}^{(k,l)} = \begin{bmatrix} \vdots \\ \sin\left((i-1)\frac{k\pi}{p}\right)\sin\left((j-1)\frac{l\pi}{q}\right) \\ \sin\left((i-1)\frac{k\pi}{p}\right)\sin\left(j\frac{l\pi}{q}\right) \\ \sin\left((i-1)\frac{k\pi}{p}\right)\sin\left((j+1)\frac{l\pi}{q}\right) \\ \vdots \\ \sin\left(i\frac{k\pi}{p}\right)\sin\left((j-1)\frac{l\pi}{q}\right) \\ \sin\left(i\frac{k\pi}{p}\right)\sin\left(j\frac{l\pi}{q}\right) \\ \sin\left(i\frac{k\pi}{p}\right)\sin\left((j+1)\frac{l\pi}{q}\right) \\ \vdots \\ \sin\left((i+1)\frac{k\pi}{p}\right)\sin\left((j-1)\frac{l\pi}{q}\right) \\ \sin\left((i+1)\frac{k\pi}{p}\right)\sin\left(j\frac{l\pi}{q}\right) \\ \sin\left((i+1)\frac{k\pi}{p}\right)\sin\left((j+1)\frac{l\pi}{q}\right) \\ \vdots \end{bmatrix},$$

where $1 \leq i, k \leq p-1$, and $1 \leq j, l \leq q-1$. Here, k and l are the wavenumbers or frequency numbers that represent the number of half sine waves represented along the x and y directions, respectively, to constitute the eigenvectors. Smaller k or l values give long, smooth waves in a direction and larger k or l values give highly oscillatory waves. If $1 \leq k < \frac{p}{2}$ and $1 \leq l < \frac{q}{2}$, we say that the mode $\boldsymbol{u}^{(k,l)}$ is smooth, and if $\frac{p}{2} \leq k \leq p-1$ or $\frac{q}{2} \leq l \leq q-1$, we call the mode oscillatory. We consider solving $A\boldsymbol{v} = \boldsymbol{f}$ using $\boldsymbol{u}^{(k,l)}$ as our initial approximation.

Writing $\boldsymbol{u}$ to denote an approximation to the exact solution generated by the

iterative method, the error in the approximation is given by $\boldsymbol{v} - \boldsymbol{u}$. Note that the error we are talking about is the numerical error in the solution of (1.5), and not the discretization error, $\boldsymbol{V} - \boldsymbol{v}$ or the total error, $\boldsymbol{V} - \boldsymbol{u}$, where $\boldsymbol{V}$ is the exact solution of (1.2). Following [16], we have the relationship between the error in the m-th iteration, $\boldsymbol{e}^{(m)}$ and the error in the initial approximation, $\boldsymbol{e}^{(0)}$:

$$\boldsymbol{e}^{(m)} = [(D - L)^{-1}U]^m \boldsymbol{e}^{(0)} = R^m \boldsymbol{e}^{(0)}. \tag{1.7}$$

Though the eigenvectors of the iteration matrix, R^m, are not the Fourier modes, we consider the initial error consisting of Fourier modes that are also the eigenvectors of the discretization matrix, A. We start with the initial approximations $\boldsymbol{u}^{(1,1)}$, $\boldsymbol{u}^{(5,5)}$, $\boldsymbol{u}^{(10,10)}$, and $\boldsymbol{u}^{(28,28)}$ and apply the Gauss-Seidel iteration 100 times to Eq. (1.5) on a 32×32 mesh.

We plot the log of the maximum norm of the error against the iteration number for $k = l = 1, 5, 10$, and 28 in Figure 1.1. We see an initial linear decrease in the log of the error norm for all k and l and that the decrease for $k = l = 28$ is faster than that for $k = l = 1, 5$, or 10, indicating that the iterative method applied here is more effective for oscillatory waves. To confirm this, we now consider the initial approximation vector

$$\boldsymbol{u} = \frac{1}{4}(\boldsymbol{u}^{(1,1)} + \boldsymbol{u}^{(5,5)} + \boldsymbol{u}^{(10,10)} + \boldsymbol{u}^{(28,28)})$$

and apply the Gauss-Seidel iteration on 32×32, 64×64, and 128×128 meshes. Again, we plot the maximum norm of the error against the number of iterations in Figure 1.2. We see that initially the rate of decrease is faster; however, as the number of iterations increases, the rate of decrease becomes slower. Also, the decrease in error gets worse as the mesh size becomes larger. This is because, at the beginning, there are both smooth and oscillatory modes of the error and the oscillatory modes are eliminated quickly. Note, the rate for $\boldsymbol{u}^{(1,1)}$ defines asymptotic rate in Figure 1.1. After the elimination of the oscillatory modes, the smooth modes are left. As standard iterative methods are not effective at eliminating the low frequency modes, the decrease in error becomes slower as the iteration continues. This property of quickly eliminating the oscillatory error modes and leaving the smooth modes is called a smoothing property that is possessed by many conventional relaxation methods. It is also obvious that the iterative method Jacobi, weighted Jacobi, Gauss-Seidel, and successive over-relaxation are not efficient for ill-conditioned problems. Hence, solving ill-conditioned large systems to higher accuracy in numerical models requires more

efficient approaches. One efficient approach is to use multigrid (MG) or domain decomposition (DD) methods to solve the systems and to employ the standard iterative methods as preconditioners. The most efficient approach probably is to use the MG or DD methods as preconditioners for Krylov methods.

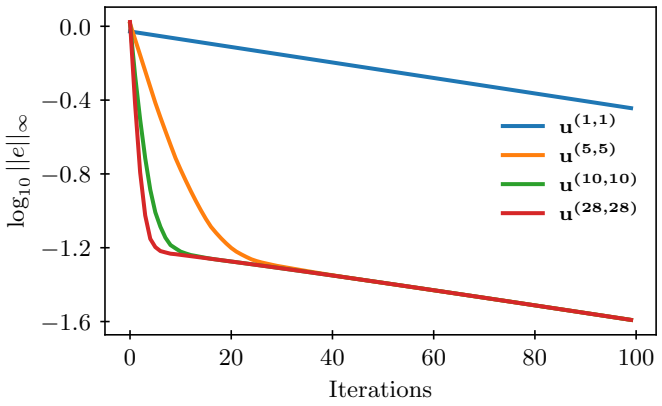


Figure 1.1: Change in $\log_{10}||e||_\infty$ with the number of iterations for the Gauss-Seidel iteration applied to Eq. (1.5) on a 32×32 mesh for 100 iterations. $\boldsymbol{u}^{(1,1)}$, $\boldsymbol{u}^{(5,5)}$, $\boldsymbol{u}^{(10,10)}$, and $\boldsymbol{u}^{(28,28)}$ are taken as the initial guesses.

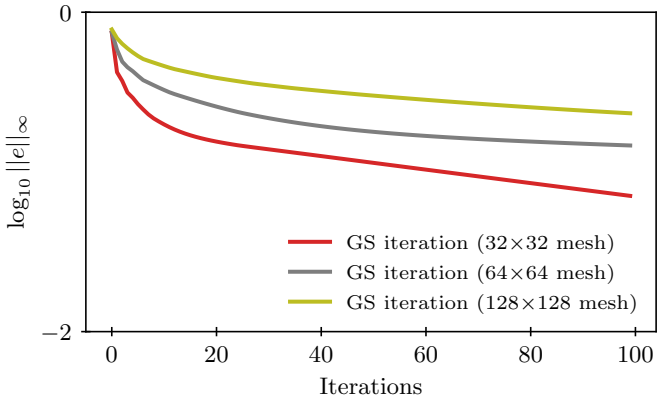


Figure 1.2: Change in $\log||e||_\infty$ with the number of iterations for the Gauss-Seidel iteration applied to Eq. (1.5) on 32×32, 64×64, and 128×128 mesh and an initial guess $\frac{1}{4}(\boldsymbol{u}^{(1,1)} + \boldsymbol{u}^{(5,5)} + \boldsymbol{u}^{(10,10)} + \boldsymbol{u}^{(28,28)})$ for 100 iterations.

Geometric multigrid methods were developed and applied to the linear systems arising from discretization of elliptic PDEs on structured grids. For the problems

on unstructured grids or with discontinuous coefficients, geometric multigrid is not suitable; algebraic multigrid method (AMG) was developed to handle these cases. Further development of algebraic multigrid methods is the main focus of this thesis.

1.4 Literature review

Multigrid methods were first studied by Fedorenko [24, 25] and then by Bakhvalov [2]. However, Brandt [10, 11] first recognized the true strength and potential efficiency of multigrid methods. Initially unaware of this work, Hackbusch also developed fundamental elements of multigrid methods [28]. These original developments were for geometric multigrid methods, which depend on detailed knowledge of the PDE to be solved and the grid on which it is discretized. To remove these limitations, classical AMG was first introduced by Brandt et al. [8, 9] and the method is now widely used. AMG is also implemented in different libraries like hypre [23], PETSc [3, 4], Trilinos [26, 30], and PyAMG [38].

There are many variants of the AMG methodology. The common features among all the variants are the construction of the coarse grid from the given fine-grid discretization matrix, and the algebraic interpolation operator. However, the construction of the coarse grid is different in different variants. Coarsening in AMG can be classified as classical coarsening, aggregation-based coarsening, and coarsening in AMGr.

1.4.1 Classical coarsening

Classical coarsening approach in AMG, also called Ruge-Stüben (RS) coarsening, was developed by Brandt, McCormick, and Ruge [8, 9] and later by Ruge and Stüben [41]. In classical AMG [9, 41], the coarse grid is a subset of the fine-grid degrees of freedom (points) and they are selected based on the idea of strength of connection in the graph of the matrix representing connections between the fine-level variables. To form the coarse-grid set, a maximal independent set is first constructed from the graph of strong connections between the fine-grid degrees of freedom and then the coarse-grid set is augmented to ensure the construction of suitable interpolation from the final coarse grid. This can be accomplished using a greedy algorithm. The classical coarsening algorithm is sequential, however, there are many alternative strategies to overcome the inherent sequential nature of the algorithm [1, 20, 21, 22, 29]. Also, different

approaches have been developed for coarsening, for example, based on redefining the notion of strength of connection [12, 17, 19, 39] or use of compatible relaxation principles [14, 32]. Once the coarse grid has been constructed, an interpolation operator is determined based on the given matrix entries in A and the principle that errors after relaxation vary smoothly between strongly connected points. Figure 1.3 shows the coarse-grid and fine-grid points for the bilinear finite-element discretization of an isotropic (left) and an anisotropic (right) problem on a uniform 32×32 mesh from classical coarsening.

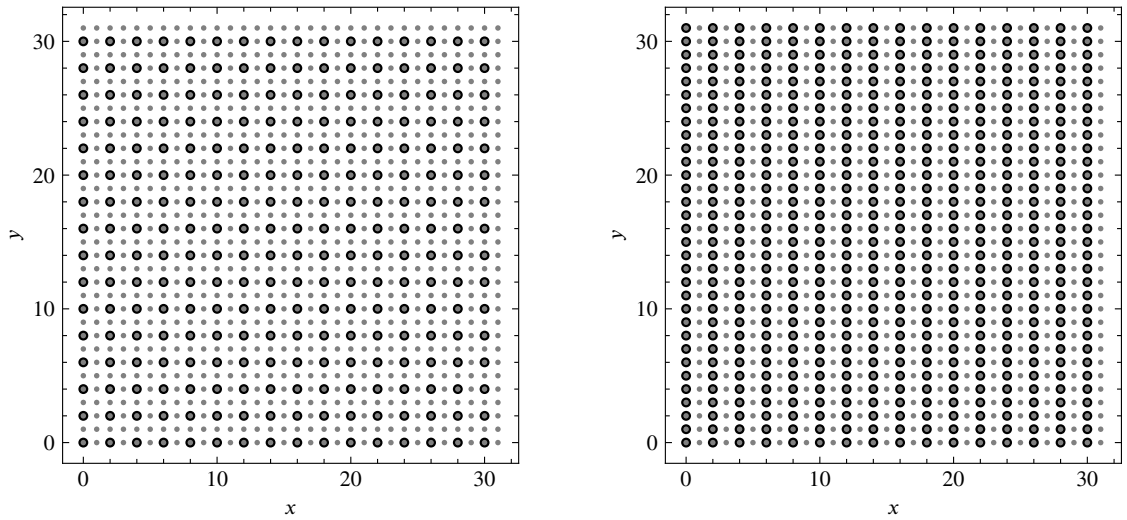


Figure 1.3: Splitting of C- and F- points for uniform 32×32 meshes from classical Ruge-Stüben (RS) coarsening. At left, coarsening for the FE discretization of isotropic-diffusion. At right, coarsening for the FE discretization of anisotropic-diffusion with strength of anisotropy 10^{-6} and direction of anisotropy $\pi/3$. Fine-grid DoFs are denoted by filled grey dots; those that are in C are marked with black circles.

1.4.2 Aggregation-based coarsening

In aggregation-based approaches, coarse grids are defined by clustering fine-grid points into aggregates. The DoFs, $\{i\}_{i=1}^{N_{\text{node}}}$, that are also the fine nodes in the first level, are partitioned and grouped into disjoint aggregates, $\{C_a\}_{a=1}^{N_{\text{cluster}}}$. Aggregates can be formed using an aggregation method, i.e., greedy aggregation [36], MIS(2) aggrega-

tion [5], or Metis aggregation [31]. As examples, Figure 1.4 shows greedy coarsening for the bilinear finite-element discretization of an isotropic (left) and an anisotropic (right) problem on a uniform 32×32 mesh. Each aggregate represents a node in the coarse grid. In this setting, the coarse nodes don't need to have any geometric meaning. In the aggregation setting, interpolation is determined by using the aggregates to define a partition of unity that is applied to a number of vectors that represent so-called algebraically smooth error. Smoothed aggregation approaches [15, 44, 45, 46] then postprocess this "tentative interpolation" operator by applying a local relaxation scheme column-wise, to yield improved interpolation of slow-to-converge modes. Aggregation-based methods have been studied for a number of years [7, 18, 36], but are not as widely used because they may not yield methods that are as efficient as the point-based coarsening in classical AMG in some settings [42, 47].

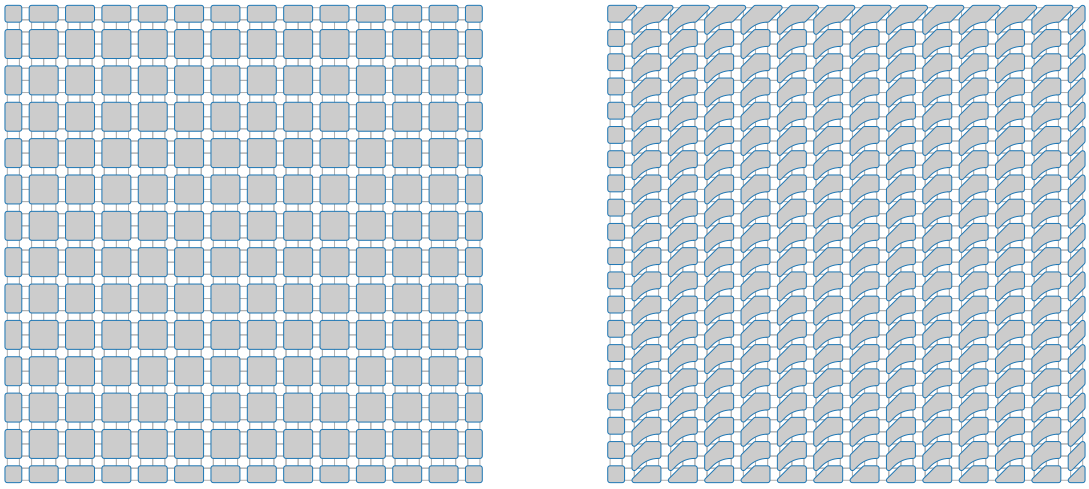


Figure 1.4: Example of greedy aggregation for uniform 32×32 meshes. At left, aggregation for the FE discretization of isotropic-diffusion. At right, aggregation for the FE discretization of anisotropic-diffusion with strength of anisotropy 10^{-6} and direction of anisotropy $\pi/3$.

1.4.3 AMGr

Classical AMG heuristics generally fail to provide robust solvers for difficult classes of problems, such as the Helmholtz equation, convection-dominated flows, or coupled

systems of PDEs. One approach to try to generalize AMG for broader classes of problems is to abandon the heuristics and to tie the algorithms to multigrid convergence theory. In this approach, there is one class of method that couples pairwise aggregation with suitable relaxation and provides convergence guarantees for symmetric, diagonally dominant M-matrices [13, 37]. The second class of methods in this approach are reduction-based AMG methods [27, 33, 34, 35]. Ries et al. [40] first proposed the reduction-based multigrid method as a generalization of cyclic reduction [43]. The idea of reduction-based multigrid methods comes from the recognition that cyclic reduction can be interpreted as a two-level multigrid method. The focus of this class of algorithm is, then, a direct connection between properties of the fine-grid submatrix and the guaranteed convergence rate of the scheme [27, 33, 34, 35]. AMGr, a reduction-based multigrid method, was proposed in [34]. AMGr gives a theoretical framework to analyse convergence based on diagonal dominance of the fine-grid submatrix (in contrast to the fine-grid submatrix being diagonal in the classical setting of cyclic reduction). In particular, given a partitioning of the DoFs into F and C subsets, AMGr defines interpolation based on approximating the so-called "ideal" interpolation operator using a diagonal (or some other easily inverted) approximation to the fine-grid submatrix, and a corresponding F-relaxation scheme. The key challenge in determining such a partitioning was studied in [35], and discussed in detail later in this thesis. The method was extended to Chebyshev-style relaxation schemes in [27]. In [33, 35], this approach was used to guide construction of a family of greedy coarsening algorithms that aim to maximize the size of the fine-grid set subject to maintaining a fixed level of diagonal dominance in the fine-grid submatrix (and, consequently, the resulting convergence bound on the AMG method).

1.5 Contributions of this Thesis

Coarse grids for AMGr can be determined from the solution of the NP-complete integer linear programming problem formulated by MacLachlan and Saad [33, 35]. Their formulation aims to minimize the size of the coarse grid while maintaining a minimum level of diagonal dominance in the fine-grid submatrix and consequently to make the resulting method more efficient. MacLachlan and Saad [35] proposed a greedy algorithm to solve the problem. In this thesis, we revisit the coarsening process in classical AMGr and develop a new coarsening algorithm based on simulated annealing (SA) following the theory and framework for AMGr [34, 35]. This SA approach

to coarsening is shown to improve on the greedy coarsening, yielding smaller coarse grids without sacrificing convergence guarantees. The smaller coarse-grid problem yields moderately more efficient (two-level) algorithms.

The AMGr methodology is important to consider because this method can provide guarantees on convergence rates, even when applied to problems that are not diagonally dominant. However, classical AMGr using the greedy or SA coarsening approaches, does not provide a guaranteed convergence rate if the coefficient matrix is not diagonally dominant. In the second part of this thesis, we take the challenge to generalize and improve the classical AMGr approach and to make the new method better applicable to problems that are not diagonally dominant. We improve classical AMGr by using SPAI to build better relaxation and interpolation operators. We truncate the less-important elements from the interpolation operator to reduce the computational cost while retaining the interpolation accuracy. The accuracy and efficiency of the interpolation operator is further improved by employing an "improved iteration" technique. Finally, we include "C-relaxation" to get better convergence factors. The new SPAI-based algorithm can be applied to problems that are diagonally dominant or not and this new method is more efficient compared to the classical AMGr method. Numerical results are presented to show the operator complexities and convergence factors for the new method.

Clustering or aggregation is an essential task in aggregation based multigrid methods. While using existing aggregation methods, we observed that the aggregates or clusters formed are often either not uniform, well-rounded, well-shaped, and well-centered or the number of clusters cannot be effectively controlled. This observation also raises the question if the quality of the aggregates has any effect on the resulting AMG convergence factor. To answer this question, we address the shortcomings in aggregation-based multigrid methods by introducing improvements to Lloyd aggregation and studying their effectiveness. The new algorithm yields aggregates that are seen to be more uniform in size, well-rounded, well-shaped, and well-centered. The number of clusters can also be directly controlled. The new algorithm exceeds the existing algorithms in performance. We also observe that the quality of the aggregates influence the performance of the aggregation-based AMG methods. We present theoretical results to establish linear complexity of the clustering algorithm and provide numerical results to show the performance of the improved clustering.

1.6 Outline

The remainder of the thesis is organized as follows:

Chapter 2 reviews the fundamental components of multigrid, algebraic multigrid, and reduction-based algebraic multigrid methods.

Chapter 3 presents a new coarsening algorithm to use in reduction-based multigrid. Here, a constrained combinatorial optimization problem is solved using simulated annealing to develop the algorithm. We show that the new coarsening algorithm performs better than the existing greedy coarsening algorithm. Numerical results are presented to show the performance of the new coarsening algorithm.

Chapter 4 describes the improved classical AMGr method. We generalize the classical AMGr method to improve its performance on matrices that are not diagonally dominant. A sparse approximate inverse method has been employed to form better relaxation and interpolation operators. We present numerical results showing the robustness of the new method for non-diagonally dominant systems.

Chapter 5 introduces a new algorithm to form balanced aggregates for the graph of a sparse matrix by introducing improvements to Lloyd aggregation. Here, we provide theoretical results to establish linear complexity. Numerical results in the context of algebraic multigrid are presented to show the performance of the improved clustering.

Bibliography

[1] David M. Alber and Luke N. Olson. Parallel coarse-grid selection. *Numerical Linear Algebra with Applications*, 14(8):611–643, 2007. URL `http://dx.doi.org/10.1002/nla.541`.

[2] Nikolai Sergeevich Bakhvalov. On the convergence of a relaxation method with natural constraints on the elliptic operator. *USSR Computational Mathematics and Mathematical Physics*, 6(5):101–135, 1966.

[3] Satish Balay, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.0.0, Argonne National Laboratory, 2008.

[4] Satish Balay, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G.

Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page. Technical report, 2010. http://www.mcs.anl.gov/petsc.

[5] Nathan Bell, Steven Dalton, and Luke N Olson. Exposing fine-grained parallelism in algebraic multigrid methods. *SIAM Journal on Scientific Computing*, 34(4): C123–C152, 2012.

[6] Garrett Birkhoff and JA George. Elimination by nested dissection, complexity of sequential and parallel numerical algorithms, 1973.

[7] Dietrich Braess. Towards algebraic multigrid for elliptic problems of second order. *Computing*, 55(4):379–393, 1995.

[8] A. Brandt, S. F. McCormick, and J. W. Ruge. Algebraic multigrid (AMG) for automatic multigrid solutions with application to geodetic computations. Technical report, Inst. for Computational Studies, Fort Collins, Colo., 1982.

[9] A. Brandt, S. F. McCormick, and J. W. Ruge. Algebraic multigrid (AMG) for sparse matrix equations. In D. J. Evans, editor, *Sparsity and Its Applications*, pages 257–284. Cambridge University Press, Cambridge, 1984.

[10] Achi Brandt. Multi-level adaptive technique (MLAT) for fast numerical solution to boundary value problems. In *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics*, pages 82–89. Springer, 1973.

[11] Achi Brandt. Multi-level adaptive techniques (MLAT) for partial differential equations: Ideas and software. In *Mathematical Software*, pages 277–318. Elsevier, 1977.

[12] J. Brannick, M. Brezina, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge. An energy-based AMG coarsening strategy. *Numer. Linear Alg. Appl.*, 13:133–148, 2006.

[13] J. Brannick, Y. Chen, J. Kraus, and L. Zikatanov. Algebraic multilevel preconditioners for the graph Laplacian based on matching in graphs. *SIAM Journal on Numerical Analysis*, 51(3):1805–1827, 2013. doi: 10.1137/120876083.

[14] James J. Brannick and Robert D. Falgout. Compatible relaxation and coarsening in algebraic multigrid. *SIAM Journal on Scientific Computing*, 32(3):1393–1416, 2010.

[15] Marian Brezina and Petr Vaněk. A black-box iterative solver based on a two-level schwarz method. *Computing*, 63(3):233–263, 1999.

[16] William L Briggs, Van Emden Henson, and Steve F McCormick. *A multigrid tutorial*. SIAM, 2000.

[17] Oliver Bröker. *Parallel Multigrid Methods using Sparse Approximate Inverses*. Ph.D. thesis, Swiss Federal Institute of Technology, Zurich, Zurich, Switzerland, 2003.

[18] VE Bulgakov. Multi-level iterative technique and aggregation concept with semi-analytical preconditioning for solving boundary-value problems. *Communications in Numerical Methods in Engineering*, 9(8):649–657, 1993.

[19] Edmond Chow. An unstructured multigrid method based on geometric smoothness. *Numer. Linear Alg. Apps.*, 10:401–421, 2003.

[20] A. J. Cleary, R. D. Falgout, V. E. Henson, and J. E. Jones. Coarse-grid selection for parallel algebraic multigrid. In *Proc. of the Fifth International Symposium on Solving Irregularly Structured Problems in Parallel*, volume 1457 of *Lecture Notes in Computer Science*, pages 104–115. Springer–Verlag, New York, 1998.

[21] Hans De Sterck, Ulrike Meier Yang, and Jeffrey J. Heys. Reducing complexity in parallel algebraic multigrid preconditioners. *SIAM J. Matrix Anal. Appl.*, 27 (4):1019–1039, 2006. ISSN 0895-4798.

[22] Hans De Sterck, Robert D. Falgout, Joshua W. Nolting, and Ulrike Meier Yang. Distance-two interpolation for parallel algebraic multigrid. *Numer. Linear Algebra Appl.*, 15(2-3):115–139, 2008. ISSN 1070-5325.

[23] R.D. Falgout and U.M. Yang. Hypre: A library of high performance preconditioners. In *Computational Science - ICCS 2002: International Conference, Amsterdam, The Netherlands, April 21-24, 2002. Proceedings, Part III*, volume 2331 of *Lecture Notes in Computer Science*, pages 632–641. Springer-Verlag, 2002.

[24] Radii Petrovich Fedorenko. A relaxation method for solving elliptic difference equations. *USSR Computational Mathematics and Mathematical Physics*, 1(4): 1092–1096, 1962.

[25] Radii Petrovich Fedorenko. The speed of convergence of one iterative process. *USSR Computational Mathematics and Mathematical Physics*, 4(3):227–235, 1964.

[26] M.W. Gee, C.M. Siefert, J.J. Hu, R.S. Tuminaro, and M.G. Sala. ML 5.0 smoothed aggregation user's guide. Technical Report SAND2006-2649, Sandia National Laboratories, 2006.

[27] Florian Gossler and Reinhard Nabben. On AMG methods with F-smoothing based on Chebyshev polynomials and their relation to AMGr. *Electron. Trans. Numer. Anal.*, 45:146–159, 2016.

[28] W Hackbusch. Ein iteratives verfahren zur schnellen auflösung elliptischer randwertprobleme, rep. 76-12. *Institute for Applied Mathematics, University of Cologne, West Germany, Cologne*, pages 77–8, 1976.

[29] V.E. Henson and U.M. Yang. BoomerAMG: A parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, 41:155–177, 2002.

[30] Michael A. Heroux, Roscoe A. Bartlett, Vicki E. Howle, Robert J. Hoekstra, Jonathan J. Hu, Tamara G. Kolda, Richard B. Lehoucq, Kevin R. Long, Roger P. Pawlowski, Eric T. Phipps, Andrew G. Salinger, Heidi K. Thornquist, Ray S. Tuminaro, James M. Willenbring, Alan Williams, and Kendall S. Stanley. An overview of the Trilinos project. *ACM Trans. Math. Softw.*, 31(3):397–423, 2005. ISSN 0098-3500. doi: http://doi.acm.org/10.1145/1089014.1089021.

[31] George Karypis. Metis: Unstructured graph partitioning and sparse matrix ordering system. *Technical Report*, 1997.

[32] O. Livne. Coarsening by compatible relaxation. *Numer. Linear Alg. Appl.*, 11 (2):205–227, 2004.

[33] S. MacLachlan and Y. Saad. Greedy coarsening strategies for nonsymmetric problems. *SIAM J. Sci. Comput.*, 29(5):2115–2143, 2007.

[34] S. MacLachlan, T. Manteuffel, and S. McCormick. Adaptive reduction-based AMG. *Numer. Linear Alg. Appl.*, 13:599–620, 2006.

[35] Scott MacLachlan and Yousef Saad. A greedy strategy for coarse-grid selection. *SIAM Journal on Scientific Computing*, 29(5):1825–1853, 2007. doi: 10.1137/060654062.

[36] Stanislav Míka and Petr Vaněk. Acceleration of convergence of a two-level algebraic algorithm by aggregation in smoothing process. *Applications of Mathematics*, 37(5):343–356, 1992.

[37] Artem Napov and Yvan Notay. An algebraic multigrid method with guaranteed convergence rate. *SIAM Journal on Scientific Computing*, 34(2):A1079–A1109, 2012. doi: 10.1137/100818509.

[38] L. N. Olson and J. B. Schroder. PyAMG: Algebraic multigrid solvers in Python v4.0. Technical report, 2018. URL `https://github.com/pyamg/pyamg`. Release 4.0.

[39] Luke N Olson, Jacob Schroder, and Raymond S Tuminaro. A new perspective on strength measures in algebraic multigrid. *Numerical Linear Algebra with Applications*, 17(4):713–733, 2010.

[40] M. Ries, U. Trottenberg, and G. Winter. A note on MGR methods. *J. Lin. Alg. Applic.*, 49:1–26, 1983.

[41] J. W. Ruge and K. Stüben. Algebraic multigrid (AMG). In S. F. McCormick, editor, *Multigrid Methods*, volume 3 of *Frontiers in Applied Mathematics*, pages 73–130. SIAM, Philadelphia, PA, 1987.

[42] K. Stüben. An introduction to algebraic multigrid. In U. Trottenberg, C. Oosterlee, and A. Schüller, editors, *Multigrid*, pages 413–528. Academic Press, London, 2001.

[43] Paul N. Swarztrauber. The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle. *SIAM Rev.*, 19(3):490–501, 1977. ISSN 0036-1445. doi: 10.1137/1019071.

[44] P Vaněk, J Mandel, and M Brezina. Algebraic multigrid based on smoothed aggregation for second and fourth order problems. *Computing*, 56:179–196, 1996.

[45] P Vaněk, M Brezina, and J Mandel. Convergence of algebraic multigrid based on smoothed aggregation. *Numerische Mathematik*, 88(3):559–579, 2001.

[46] Petr Vaněk. Acceleration of convergence of two-level algorithm by smoothing transfer operator. *Applic. Math.*, 37:265–274, 1992.

[47] Roman Wienands and Cornelis W Oosterlee. On three-grid fourier analysis for multigrid. *SIAM Journal on Scientific Computing*, 23(2):651–671, 2001.

Chapter 2

Background

The following chapters of this thesis detail the development of new tools for selection of coarse grids in AMG methods (Chapter 3 and 5) and interpolation and relaxation operators in AMGr (Chapter 4). Here, we present background material needed for this later work. In particular, we follow [10] and introduce the standard geometric multigrid algorithm for the two-dimensional Poisson problem, highlighting some of the basic principles of the multigrid methodology. With these, we introduce specific material on aggregation methods used in AMG (Section 2.3) and give a brief description of the classical AMGr method (Section 2.4).

2.1 (Geometric) multigrid

Multigrid methods are well-known to be effective algorithms for the solution of large linear systems of equations arising in many areas of computational science and engineering [10, 24]. They can perform as stand-alone solvers for many problems, but are also effective as preconditioners for Krylov methods. Multigrid methods originate with the observation that the error in an approximation to the solution of a system of equations can be decomposed into its high-frequency and low-frequency modes. Multigrid methods arise from the observation that standard iterative methods, such as the (weighted) Jacobi and Gauss-Seidel iterations, can effectively eliminate high-frequency or oscillatory components of the error. These methods cannot, however, effectively eliminate the low-frequency or smooth components. Instead, the smooth components can be corrected from a "coarse-grid" representation of the problem. The complementarity between relaxation methods, such as Jacobi and Gauss-Seidel,

and this coarse-grid correction process leads to a very effective approach to damp all modes in the system.

In a multigrid method, relaxation is first done on $A\boldsymbol{v} = \boldsymbol{f}$ on the fine grid Ω^h to obtain an approximation, $\boldsymbol{u^h}$. In the simplest case, of one-dimensional and two-dimensional (rectangular) grids, we consider a fine grid that is a uniform mesh with distance h spacing between two grid-points and a coarse-grid that has distance $2h$ spacing. Hence, the superscripts h and $2h$ are used to represent the fine and coarse grids, respectively. We assume that we know an interpolation matrix, P, that maps vectors from the coarse grid to the fine grid. In many cases, this is easily determined geometrically, as (for example) piecewise linear or bilinear interpolation, or the natural interpolation operator determined by a finite-element discretization. When relaxation begins to stall, the approximation, $\boldsymbol{u^h}$, from the relaxation method is dominated by the smooth portion of the initial error that came from the initial guess. We then form the residual, $\boldsymbol{r^h} = \boldsymbol{f} - A\boldsymbol{u^h}$, to help find the smooth error in the approximation. The residual, $\boldsymbol{r^h}$, is transferred to the coarse grid, Ω^{2h}, using an intergrid transfer operator called the restriction operator. Among several possible choices, we focus on the variational setting common in algebraic multigrid, where the restriction operator is given as P^T, and $\boldsymbol{r^{2h}} = P^T\boldsymbol{r^h}$. The discretization matrix on the coarse grid can be defined via the Galerkin condition as $A^{2h} = P^TAP$. An iterative method can be applied again to the residual equation $A^{2h}\boldsymbol{e^{2h}} = \boldsymbol{r^{2h}}$ on Ω^{2h} or this equation can be solved directly to obtain an approximation to the error $\boldsymbol{e^{2h}}$. The approximation for the error on the coarse grid is then used to represent the smooth error. Now this approximation for the error, $\boldsymbol{e^{2h}}$, found on the coarse grid is transferred to the fine grid to approximate $\boldsymbol{e^h}$ using an interpolation operator, P. The approximation $\boldsymbol{u^h}$ obtained on Ω^h is finally corrected by adding the error estimate $\boldsymbol{e^h}$. This two-grid correction scheme makes the solution on the finer grid more accurate. This scheme can be represented in terms of the following operations

$$\boldsymbol{u^h} \leftarrow R^\nu\boldsymbol{u^h} + C(\boldsymbol{f}) + P(A^{2h})^{-1}P^T(\boldsymbol{f^h} - A^h(R^\nu\boldsymbol{u^h} + C(\boldsymbol{f}))), \tag{2.1}$$

where ν is the number of pre-relaxation steps and $C(\boldsymbol{f})$ represents a series of operation on $\boldsymbol{f}$. As the solution $\boldsymbol{v^h}$ should be unchanged by the two-grid correction scheme, we can write

$$\boldsymbol{v^h} \leftarrow R^\nu\boldsymbol{v^h} + C(\boldsymbol{f}) + P(A^{2h})^{-1}P^T(\boldsymbol{f^h} - A^h(R^\nu\boldsymbol{v^h} + C(\boldsymbol{f}))). \tag{2.2}$$

Subtracting Eq. (2.1) from Eq. (2.2) we get the following relationship between the

error before and after the two-grid correction scheme,

$$\boldsymbol{e^h} \leftarrow [I - P(P^T AP)^{-1} P^T A] R^\nu \boldsymbol{e^h} \equiv CLC \cdot R^\nu \boldsymbol{e^h}, \tag{2.3}$$

where I is the identity matrix and CLC is the coarse level correction operator. Recursion of the two grid scheme results in a multigrid method.

To show the effectiveness of the multigrid method, we revisit the two-dimensional Poisson problem, Equation (1.2), and apply the method with the same initial approximation vector. The results are plotted in Figure 2.1. We see that the multigrid method eliminates both the smooth and oscillatory modes with a small number of iterations, and that the increase in problem size does not deteriorate the performance of the multigrid method.

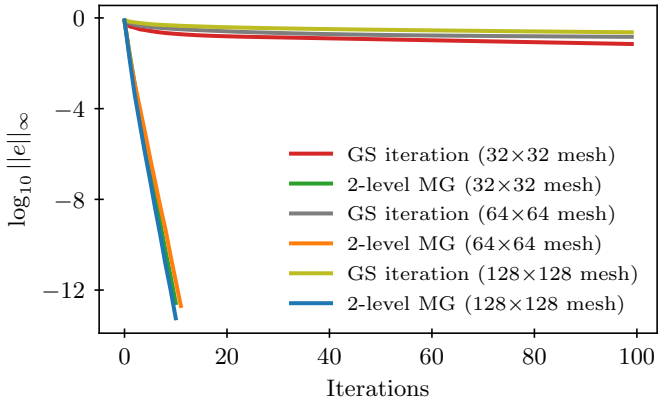


Figure 2.1: Change in $\log ||e||_\infty$ with the number of iterations for the Gauss-Seidel iteration and two-level multigrid applied to Eq. (1.5) on 32×32, 64×64, and 128×128 mesh and an initial guess $\frac{1}{4}(\boldsymbol{u^{(1,1)}} + \boldsymbol{u^{(5,5)}} + \boldsymbol{u^{(10,10)}} + \boldsymbol{u^{(28,28)}})$ for 100 iterations.

The key idea in geometric multigrid is that the smooth components of an error on a finer grid become oscillatory on a coarser grid, showing a way to solve the issue arising with convergence of conventional iterative methods. As a simple example, consider a one-dimensional grid, and let the even-numbered grid points of the fine grid define the coarse grid. In this case, the number of grid-points on the coarser grid is roughly half of that on the finer grid. If a wave on the fine grid is projected to the coarse grid, its wavenumber remains the same. As the wavenumbers on both of the fine and coarse grids are the same, but the coarse grid has fewer grid points, a smooth wave on the fine grid looks more oscillatory on the coarse grid. This allows relaxation on the coarse grid to effectively resolve some error modes that are poorly

reduced by fine-grid relaxation.

There are two broad classes of multigrid methods- geometric and algebraic. As described above, in geometric multigrid, coarse grids are chosen based on regular geometric patterns, and polynomial interpolation is used. Another class, called Black Box Multigrid (BoxMG), lies between geometric and algebraic multigrid using structured grids but not polynomial interpolation. Unlike geometric multigrid, BoxMG uses an operator-induced interpolation procedure, providing a "black box" where the user needs to provide only the fine-grid discretization, right-hand side, and an initial guess for the solution. For classical algebraic multigrid, we generalize the choice of geometric grids, requiring only the structure that there is a disjoint splitting of the fine grid: $\Omega = F \cup C$, where C is the set of the grid points on the coarse grid and F is its complement belonging only on the fine level. We thus have $\Omega^{2h} = C$ but the same cycling structure as described above. The number of grid points on the coarser grid is kept sufficiently small to reduce the cost of solving the residual problem.

Geometric multigrid can be applied when the grid locations are known and are structured or regular. In this case, all of the elements of multigrid methods come naturally. Grid points on the coarser grid are generally chosen so that the grid spacing is twice that on the next finer grid. The choice of the grid-transfer operators depends on the details of the discretization. For finite-difference discretizations, piecewise linear interpolation is often used to interpolate the error from the coarse grid to the fine grid. Error from a fine grid can be restricted to a coarser grid using injection, where the coarse-grid vector simply takes its value directly from the corresponding fine-grid point. There is an alternate restriction operator, called full weighting, where the coarse-grid vector values are given by weighted averages of values at neighboring fine-grid points. Smooth error in geometric multigrid is the component of the error that has low frequency. The coarse-grid correction scheme in geometric multigrid is very effective in eliminating smooth errors.

2.2 Algebraic multigrid

Geometric multigrid methods are difficult or impossible to apply when the domain of the problem is complex enough that any sensible discretization is too fine to serve as the coarsest grid. It becomes difficult to apply geometric multigrid when the discretization on the finest grid does not allow uniform coarsening, or when it is difficult to find a relaxation process that smoothes the error sufficiently to admit a good

coarse-grid correction using geometric interpolation. Also, for many purely discrete problems, where the problem has no geometric background, geometric multigrid is not applicable [23]. AMG methods were developed using the usual principles of multigrid methods, but intended to use only the information contained in a given discretization matrix. AMG was invented by Brandt, McCormick, and Ruge in the early 1980's [4], and Ruge and Stüben [23] were among the first to recognize it as an interesting algorithm.

In AMG, the grid points are identified with the indices of the unknown quantities, and the connections within the grid are determined by the undirected adjacency graph of the discretization matrix. Unlike geometric multigrid methods, where relaxation is often adapted to suit the problem at hand, a relaxation method is fixed in AMG. The components of the error that cannot be reduced by the relaxation method are defined to be algebraically smooth error. However, algebraically smooth errors are not necessarily geometrically smooth as in the case of anisotropic diffusion operators, or problems with highly variable coefficients. Selection of the coarse-grid points and defining the interpolation operator are done in a way so that the error not efficiently reduced by relaxation is well approximated by the interpolation operator.

Classical AMG makes use of the notion of *strong connections* in the graph corresponding to matrix A to define the coarse-fine partitioning. In the original algorithm, point i is said to be strongly connected to point j if $-A_{ij} \geq \gamma \max_{k \neq i}(-A_{ik})$ for some $\gamma \in (0, 1]$ (where the negative signs reflect the expectation that A be an M-matrix, or one where positive off-diagonal entries are not substantial). Many other definitions of strong connections have been considered in the literature, including the symmetric strength measure commonly used in smoothed aggregation AMG, where the connection between i and j is said to be strong if $|A_{ij}| \geq \gamma \sqrt{A_{ii} A_{jj}}$. Local approximations of the inverse of A and algebraically smooth error have also been used to define strong connections [7, 21], as has the concept of algebraic distances [6]. In all cases, the strength measure is used to "filter" the entries in A, and then the set C is defined by choosing a maximal independent set over the graph of strong connections. This can be characterized, for example, by the properties that each point in F is strongly dependent on at least one C-point, but no two C-points can be strongly dependent on one-another.

The standard interpolation operator in classical AMG is formulated based on the assumption that algebraically smooth errors have small residuals after relaxation. Hence, the residual equation after relaxation can be written as $Ae \approx 0$. For the i-th

degree of freedom, $i \in F$, this gives

$$A_{ii}e_i \approx -\sum_{k\in C_i} A_{ik}e_k - \sum_{j\in F_i^s} A_{ij}e_j - \sum_{\ell\in F_i^w} A_{i\ell}e_\ell, \tag{2.4}$$

where C_i, F_i^s, and F_i^w are the C-neighbours, strongly connected F-neighbours, and weakly connected F-neighbours of the i-th point, respectively. The standard interpolation operator is then derived by assuming that $e_\ell \approx e_i$ for weakly-connected neighbours, while

$$e_j \approx \frac{\sum_{k\in C_i} A_{jk}e_k}{\sum_{m\in C_i} A_{jm}},$$

for points $j \in F_i^s$. Substituting these into (2.4) and solving for e_i yields the interpolation formula

$$e_i = \sum_{k\in C_i} \frac{-A_{ik} - \sum_{j\in F_i^s} \frac{A_{ij}A_{jk}}{\sum_{m\in C_i} A_{jm}}}{A_{ii} + \sum_{\ell\in F_i^w} A_{i\ell}} e_k. \tag{2.5}$$

This defines the interpolation formula for all points $i \in F \subset \Omega$, assuming that e_k is known for all $k \in C_i$. For points $k \in C$, we use direct injection of values from coarse-grid points to their fine-grid counterparts in Ω.

Another viewpoint is to consider the decomposition of the error. In this view, we partition the error in an approximation into two pieces. One component lies in the range of interpolation and the other component lies in the A-orthogonal space to the range of the interpolation. In geometric MG, the range of interpolation mostly contains the smooth error and the A-orthogonal space mostly contains the oscillatory error. The two-grid correction operator is an orthogonal projection that has a column space orthogonal to the range of the interpolation. Hence, any error component in the range of interpolation is diminished by the two-grid correction scheme. However, the other component that mostly lies in the column space of the two-grid correction operator cannot be removed by the coarse grid. AMG methods aim to determine interpolation operators so that this component is quickly attenuated by relaxation.

In contrast to geometric multigrid, AMG requires a setup phase in which the components of the multigrid hierarchy are explicitly built. This phase uses the form of a recursion where, starting from the finest level, a coarse grid is chosen, and interpolation operator mapping from the coarse grid to the fine grid is computed, and the coarse-grid operator is determined. There are several approaches to building coarse grids for AMG. In classical AMG as described above, a subset of the fine level variables are selected to be the coarse level variables [5, 10, 24], whereas in

aggregation-based AMG, several fine-level variables are combined into a single coarse-level variable [3, 9, 26].

2.3 Smoothed aggregation AMG

Smoothed Aggregation AMG method is an improvement to the aggregation multigrid method. Hence, we will first discuss the aggregation multigrid method followed by the extension to the smoothed aggregation method. In an aggregation multigrid method, the DoFs, $\{i\}_{i=1}^{N_{\text{node}}}$, that are also the fine nodes in the first level, are partitioned and grouped into disjoint aggregates, $\{C_a\}_{a=1}^{N_{\text{cluster}}}$. Aggregates can be formed using an aggregation method, e.g., greedy aggregation, MIS(2) aggregation, or Metis aggregation. Each aggregate represents a node in the coarse grid. The coarse nodes here don't need to have any geometric meaning. The restriction operator, R, is defined for row a as

$$R_{a,i} = \begin{cases} 1 & \text{if } i \in C_a, \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

For a symmetric coefficient matrix, A, the interpolation operator and the coarse-grid matrix is defined as, $P = R^T$ and $A_c = RAP$, respectively. The two-level multigrid process is convergent for these operators [20]. The range of P contains the exact null space of the differential operator that we are considering here. However, since only the piecewise constant vectors are included in the range of P, it can't capture all modes of algebraically smooth error.

Algebraically smooth error is error that can't be reduced anymore using classical relaxation methods. This is a vector in the near null space of A. In a multigrid method, the range of the interpolation operator should capture the near null space in order to eradicate the error left after relaxation. To do that, Vaněk improved the interpolation operator from non-smoothed aggregation by smoothing the columns of the tentative interpolation operator, P, by the weighted Jacobi operator [28]. The "smoothed" aspect of smoothed aggregation comes from this smoothing of the tentative interpolation operator. The improvement of the interpolation operator significantly improves the performance over the non-smoothed aggregation multigrid method.

Vaněk et al. [26, 27] extended the smoothed aggregation method to apply it to systems of PDEs and to higher-order problems. In this extension, the vectors that

span the near null space, also referred to as candidate vectors, are split into smaller vectors over each aggregate. Each of these local distributions are normalized via the QR factorization. The tentative interpolation operator, P, is then formed by redistributing the normalized vectors to the global operator. The tentative interpolation operator is improved by smoothing its columns by weighted Jacobi method, giving the interpolation operator. Restriction and the coarse-grid matrix are then formed using the variational conditions.

A multilevel method can be constructed extending the smoothed aggregation method using the smoothed aggregation method again to solve the coarse-grid problem. Coarse-level nodes are aggregated again and the interpolation operator for this level is formed using the near null space representation of the coarse-grid matrix. A key factor in the success of aggregation-based multigrid methods is how the aggregates are formed. We next review the most common algorithms.

2.3.1 Greedy aggregation

A relatively large edge weight, $W_{i,j}$ in a fine matrix graph is deemed by SA to represent a strong connection between two nodes i and j. Hence, a natural approach is to consider a connection between the nodes i and j to be strong if $|A_{i,j}| \geq \theta\sqrt{|A_{i,i} \cdot A_{j,j}|}$, where θ is a threshold parameter. Use of $\theta = 0$ or $\theta = 0.25$ is common in practice. A matrix of strong connections, W is formed where $W_{i,j} = 1$ if the connection between the nodes i and j is strong, otherwise $W_{i,j} = 0$. Construction of W is presented in Algorithm 2.1. The matrix W is used in the greedy aggregation algorithm meaning that only the strongly connected nodes are reckoned as neighbours in the aggregation process.

Algorithm 2.1 Strength of Connection Matrix

```
1: function STRENGTH-OF-CONNECTION(A, θ)
2:     W_{i,j} ← 0 for all i = 1, ..., N_node and j = 1, ..., N_node
3:     for i, j such that A_{i,j} ≠ 0 do
4:         if |A_{i,j}| ≥ θ√|A_{i,i} · A_{j,j}| then        ▷ find strong connections
5:             W_{i,j} ← 1
6:     return W
```

Greedy aggregation (also known as "greedy clustering" or "standard aggregation")

was first introduced by Míka and Vaněk [20]. The set of DoFs (nodes), $U = \{i\}_{i=1}^{N_{\text{node}}}$, are partitioned to construct the system of aggregation classes, $\{C_a\}_{a=1}^{N_{\text{cluster}}}$, based on the following properties:

1. $\forall i \in U, \exists a$ such that $i \in C_a$.
2. If $C_a \cap C_x \neq \emptyset$, then $C_a = C_x$.
3. If $k, l \in C_a$, then $\exists i \in U$ such that $k, l \in \{k \mid A_{i,k} \neq 0\}$.

The aggregation process that we present here is a close variant of the aggregation process presented in [20]. The greedy aggregation process consists of two passes. In the first pass, each node is visited to check if the node and all of its neighbours are unaggregated. If so, the node becomes a root node and the root node with its neighbours form a new aggregate. The second pass goes through the remaining nodes that are not aggregated yet. If an unaggregated node is adjacent to an aggregate, meaning that the unaggregated node has a neighbour that is already aggregated, the unaggregated node is incorporated into that aggregate. If such a neighbouring node is not found, the unaggregated node is considered a root node and the node with its neighbours that are also unaggregated form an aggregate. When an unaggregated node is adjacent to more than one aggregate, the aggregate can be chosen in different ways. It can be chosen arbitrarily, or based on the size of the aggregates. The aggregate can also be selected based on the largest/smallest index of the aggregate or the largest/smallest index of the aggregated neighbour of the unaggregated node. The process continues until each node belongs to an aggregate. The greedy aggregation process is shown in Algorithm 2.2.

2.3.2 Maximal independent set based aggregation

The greedy algorithm is inherently serial, yet there are two immediate observations. First, any two center nodes of two (distinct) aggregates must be more than two edges apart. Second, if an unaggregated node is more than two edges from any existing center, then the node is eligible to be a center of a new aggregate. Hence, the center nodes from the greedy algorithm represent a distance-2 maximal independent set or MIS(2). This leads to the MIS(2) aggregation algorithm, where an MIS(2) over the nodes is first constructed, followed by construction of the aggregation using the MIS(2) center nodes. This has been shown to exhibit a high degree of parallelism [1]; see [1, Algorithm 5] for details.

Algorithm 2.2 Greedy aggregation. See Table 5.2 for variable definitions.

function GREEDY-AGGREGATION(W)
  $m_i \leftarrow 0$ for all $i = 1, \ldots, N_{\text{node}}$ ▷ initially all nodes are unaggregated
  $a \leftarrow 0$ ▷ first aggregate index
  for $i \leftarrow 1, \ldots, N_{\text{node}}$ **do** ▷ **first pass**
    if $m_i = 0$ and $m_j = 0$ for all j s.t. $W_{i,j} \neq 0$ **then** ▷ unaggregated
      $m_i \leftarrow a$ ▷ add i and neighbors to aggregate a
      $m_j \leftarrow a$, for all j s.t. $W_{i,j} \neq 0$
      $c_a \leftarrow i$ ▷ mark aggregate center
      $a \leftarrow a + 1$ ▷ increment aggregate index
  for $i \leftarrow 1, \ldots, N_{\text{node}}$ **do** ▷ **second pass**
    if $m_i = 0$ **then** ▷ unaggregated
      if $\exists\ j$ s.t. $W_{i,j} \neq 0$ and $m_j > 0$ **then** ▷ aggregated neighbor
        $j \leftarrow \underset{j\,:\,m_j>0}{\operatorname{argmax}} W_{i,j}$ ▷ neighbor with largest weight
        $m_i \leftarrow m_j$
      else ▷ form new aggregate
        $m_i \leftarrow a$
        for j such that $W_{i,j} \neq 0$ and $m_j = 0$ **do**
          $m_j \leftarrow a$
        $a \leftarrow a + 1$ ▷ increment aggregate index
  return m, c

Given a distance-2 maximal independent set, the aggregation process is straightforward. In the first step, the index of the aggregate representing the center is propagated to its neighbors. This continues in the second step, where the index of the aggregate is propagated to the second layer of neighbors; if there are multiple aggregates adjacent to an unaggregated node, the choice is made arbitrarily (or by index). The algorithm is shown in Algorithm 2.3.

With an appropriate ordering, the first pass of MIS-based and greedy aggregation can yield identical clusters. With only minor differences in the second pass, we expect the aggregation patterns to be similar. Indeed, the convergence factors of AMG based on these two aggregation strategies are shown to be close in practice [1, Appendix].

Algorithm 2.3 MIS(2) aggregation. See Table 5.2 for variable definitions.

1: **function** MIS(2)-AGGREGATION(W)
2: $c \leftarrow \text{MIS}(W,\ 2)$ ▷ distance-2 independent set
3: $m_i \leftarrow 0$ for $i = 1, \ldots, N_{\text{node}}$
4: $N_{\text{cluster}} \leftarrow |c|$
5: **for** $a = 1, \ldots, N_{\text{cluster}}$ **do** ▷ **pass 1:** distance-1
6: $i \leftarrow c_a$ ▷ index of center for aggregate a
7: $m_i \leftarrow a$ ▷ set aggregate number for center
8: **for** j s.t. $W_{i,j} \neq 0$ **do**
9: $m_j \leftarrow a$ ▷ set aggregate number for neighbors
10: **for** i s.t. $m_i > 0$ **do** ▷ **pass 2:** distance-2
11: **for** j s.t. $W_{i,j} \neq 0$ and $m_j = 0$ **do**
12: $m_j \leftarrow m_i$ ▷ set aggregate number for neighbors
13: **return** m, c

2.3.3 Standard Lloyd aggregation

A shortcoming of the previous two aggregation strategies is the inability to control the coarsening rate: the number of aggregates or clusters is an outcome of the algorithm, rather than an input. In contrast, Lloyd aggregation, introduced in [2], is based on an initial seeding of centers (of any length). Lloyd aggregation can be viewed as an extension of Lloyd's algorithm [14] applied to graphs, where an initial random seeding of centers yields Voronoi cells (or a set of nodes closest to each center), followed by a recentering of center locations.

A full algorithm is given in Algorithm 2.4, where a subset of N_{cluster} nodes are randomly selected as the initial centers, input as c. A standard Bellman-Ford algorithm (see Algorithm 2.5 and [11, Section 8.7]) is used to find the distance and index of the closest center; the set of points closest to each center form the initial clustering. Next, the border nodes of each cluster are selected and a modified form of the Bellman-Ford algorithm then identifies the (new) center — see Algorithm 2.6 — by selecting the node of maximum distance to the cluster boundary (with ties selected arbitrarily). The steps are repeated until the algorithm has converged or a maximum number of iterations (given as $T_{\max}$) is reached.

Algorithm 2.4 Lloyd aggregation algorithm. See Table 5.2 for variable definitions.

function LLOYD-AGGREGATION($W, c, T_{\max}$)
  $t = 0$
  repeat
    $m, d \leftarrow$ BELLMAN-FORD(W, c) ▷ find closest centers
    $c \leftarrow$ MOST-INTERIOR-NODES(W, m) ▷ recenter
    $t = t + 1$
  until $t = T_{\max}$ or no change in c and m
  return m, c

Algorithm 2.5 Bellman-Ford algorithm to compute distance and index of closest center. See Table 5.2 for variable definitions.

function BELLMAN-FORD(W, c)
  $d_i \leftarrow \infty$ for all $i = 1, \ldots, N_{\text{node}}$ ▷ initial distance
  $m_i \leftarrow 0$ for all $i = 1, \ldots, N_{\text{node}}$ ▷ initial membership undefined
  for $a \leftarrow 1, \ldots, N_{\text{cluster}}$ **do**
    $i \leftarrow c_a$ ▷ cluster a has center node i
    $d_i \leftarrow 0$ ▷ distance of a center node to itself is zero
    $m_i \leftarrow a$ ▷ center node i belongs to its own cluster
  repeat
    done $\leftarrow$ true
    for i, j such that $W_{i,j} > 0$ **do** ▷ all pairs of adjacent nodes
      if $d_j + W_{i,j} < d_i$ **then** ▷ found a shorter distance to node j's center
        $m_j \leftarrow m_i$ ▷ switch node j to the same cluster as i
        $d_j \leftarrow d_i + W_{i,j}$ ▷ use the shorter distance via node i
        done $\leftarrow$ false ▷ change was made; do not terminate
  until done
  return m, d

2.4 Classical AMGr

Cyclic reduction [25] was originally proposed as a direct solver for certain linear systems that arose from finite-difference discretization of simple PDEs. Assuming that the degrees of freedom are already partitioned into coarse and fine nodes, the linear system $A\boldsymbol{v} = \boldsymbol{f}$ is reordered to have F degrees of freedom followed by C degrees

Algorithm 2.6 Find the most-interior node (furthest from boundary) for each cluster. See Table 5.2 for variable definitions.

function MOST-INTERIOR-NODES(W, m)
  $B \leftarrow \{\}$ ▷ border nodes
  for i, j such that $W_{i,j} > 0$ **do** ▷ all pairs of adjacent nodes
    if $m_i \neq m_j$ **then** ▷ are nodes i and j in different clusters?
      $B \leftarrow B \cup \{i, j\}$ ▷ if so, add both of them to the border set
  $\cdot, d \leftarrow$ BELLMAN-FORD(W, B) ▷ d is distance from cluster borders
  for $i \leftarrow 1, \ldots, N_{\text{node}}$ **do**
    $a \leftarrow m_i$ ▷ a is the cluster index for node i
    $c_a \leftarrow i$ ▷ assign the highest-index node as cluster center
  for $i \leftarrow 1, \ldots, N_{\text{node}}$ **do**
    $a \leftarrow m_i$ ▷ a is the cluster index for node i
    $j \leftarrow c_a$ ▷ j is the current cluster center
    if $d_i > d_j$ **then** ▷ is node i further from the border than j?
      $c_a \leftarrow i$ ▷ if so, node i is the new cluster center
  return c

of freedom, writing

$$A = \begin{bmatrix} A_{FF} & -A_{FC} \\ -A_{CF} & A_{CC} \end{bmatrix} \quad \boldsymbol{v} = \begin{bmatrix} \boldsymbol{v}_F \\ \boldsymbol{v}_C \end{bmatrix} \quad \boldsymbol{f} = \begin{bmatrix} \boldsymbol{f}_F \\ \boldsymbol{f}_C \end{bmatrix}. \tag{2.7}$$

An *exact* algorithm for the solution of $A\boldsymbol{v} = \boldsymbol{f}$ in this partitioned form is given by

1. $\boldsymbol{y}_F = A_{FF}^{-1}\boldsymbol{f}_F$,
2. Solve $\left(A_{CC} - A_{CF}A_{FF}^{-1}A_{FC}\right)\boldsymbol{v}_C = \boldsymbol{f}_C + A_{CF}\boldsymbol{y}_F$,
3. $\boldsymbol{v}_F = \boldsymbol{y}_F + A_{FF}^{-1}A_{FC}\boldsymbol{v}_C$.

This can be turned into an iterative method for solving $A\boldsymbol{v} = \boldsymbol{f}$ in the usual way, replacing the right-hand side vector, $\boldsymbol{f}$, by the evolving residual, and solving for an error correction, leading to reduction-based multigrid [22]. In this form, we compute updates to the current approximation, $\boldsymbol{v}^{(k)}$, as

1. $\boldsymbol{v}_F^{(k+1/2)} = \boldsymbol{v}_F^{(k)} + A_{FF}^{-1}\left(\boldsymbol{f}_F - A_{FF}\boldsymbol{v}_F^{(k)} + A_{FC}\boldsymbol{v}_C^{(k)}\right)$,
2. Solve $\left(A_{CC} - A_{CF}A_{FF}^{-1}A_{FC}\right)\boldsymbol{y}_C = \boldsymbol{f}_C + A_{CF}\boldsymbol{v}_F^{(k+1/2)} - A_{CC}\boldsymbol{v}_C^{(k)}$,

3. $\boldsymbol{v}_C^{(k+1)} = \boldsymbol{v}_C^{(k)} + \boldsymbol{y}_C$,

4. $\boldsymbol{v}_F^{(k+1)} = \boldsymbol{v}_F^{(k+1/2)} + A_{FF}^{-1} A_{FC} \boldsymbol{y}_C$.

We note that, as given above, this is still an exact algorithm, as $\boldsymbol{v}^{(1)} = \boldsymbol{v} = A^{-1}\boldsymbol{f}$ for any initial guess, $\boldsymbol{v}^{(0)}$. To make an iterative method from this skeleton, we introduce approximations of A_{FF}^{-1} in three places in the above algorithm, namely

$$\tilde{A}_{FF}^{-1} \approx A_{FF}^{-1} \qquad \tilde{A}_C \approx A_{CC} - A_{CF} A_{FF}^{-1} A_{FC} \qquad W_{FC} \approx A_{FF}^{-1} A_{FC}, \tag{2.8}$$

leading to the iteration:

1. $\boldsymbol{v}_F^{(k+1/2)} = \boldsymbol{v}_F^{(k)} + \tilde{A}_{FF}^{-1} \left(\boldsymbol{f}_F - A_{FF} \boldsymbol{v}_F^{(k)} + A_{FC} \boldsymbol{v}_C^{(k)} \right)$,

2. Solve $\tilde{A}_C \boldsymbol{y}_C = \boldsymbol{f}_C + A_{CF} \boldsymbol{v}_F^{(k+1/2)} - A_{CC} \boldsymbol{v}_C^{(k)}$,

3. $\boldsymbol{v}_C^{(k+1)} = \boldsymbol{v}_C^{(k)} + \boldsymbol{y}_C$,

4. $\boldsymbol{v}_F^{(k+1)} = \boldsymbol{v}_F^{(k+1/2)} + W_{FC} \boldsymbol{y}_C$.

Viewing this as a two-grid algorithm, we recognize the first step as special form of relaxation, known as F-relaxation, where the approximation to A_{FF}^{-1} is accomplished via a standard weighted Jacobi or Gauss-Seidel iteration. The second step then represents a coarse-grid solve, where the residual is restricted to the coarse-grid by injection, and the correction, $\boldsymbol{y}_C$, is computed using an approximation, $\tilde{A}_C$, of the true Schur complement, $A_{CC} - A_{CF} A_{FF}^{-1} A_{FC}$. The final two steps represent the interpolation of the correction, writing the interpolation operator $P = \left[\begin{smallmatrix} W_{FC} \\ I \end{smallmatrix} \right]$. This can be viewed as an approximation of the *ideal* interpolation operator [12], $W_{FC} \approx A_{FF}^{-1} A_{FC}$. Notably, this algorithm differs from standard multigrid cycling in several ways, including the fixed use of injection for the restriction of the residual to the coarse grid, and the lack of post-relaxation sweeps.

As written above, there is little guidance in how to choose the three approximations in (2.8). MacLachlan et al. [15] address this in their development of the reduction-based AMG (AMGr) algorithm, connecting convergence of the two-grid scheme with properties of A_{FF}. In particular, it is assumed that A_{FF} can be approximated by known matrix D_{FF} for which computing the action of D_{FF}^{-1} on a vector is computationally feasible. From this, Theorem 2.1 holds, using the notation that matrices $A \succeq B$ when $\boldsymbol{v}^T A \boldsymbol{v} \geq \boldsymbol{v}^T B \boldsymbol{v}$ for all vectors $\boldsymbol{v}$.

Theorem 2.1. *[15] Consider the symmetric and positive-definite matrix* $A = \begin{bmatrix} A_{FF} & -A_{FC} \\ -A_{FC}^T & A_{CC} \end{bmatrix}$ *such that* $A_{FF} = D_{FF} + \mathcal{E}$, *with* D_{FF} *symmetric,* $0 \preceq \mathcal{E} \preceq \epsilon D_{FF}$ *for some* $\epsilon \geq 0$, *and* $\begin{bmatrix} D_{FF} & -A_{FC} \\ -A_{FC}^T & A_{CC} \end{bmatrix} \succeq 0$. *Define relaxation with error-propagation operator* $R = \left(I - \sigma \begin{bmatrix} D_{FF}^{-1} & 0 \\ 0 & 0 \end{bmatrix} A\right)$ *for* $\sigma = 2/(2+\epsilon)$, *interpolation* $P = \begin{bmatrix} D_{FF}^{-1} A_{FC} \\ I \end{bmatrix}$, *and coarse-level correction with error-propagation operator* $T = I - P(P^T A P)^{-1} P^T A$. *Then the multigrid cycle with* ν *pre-relaxation sweeps, coarse-level correction, and* ν *post-relaxation sweeps has error propagation operator* $MG_2 = R^\nu \cdot T \cdot R^\nu$ *which satisfies*

$$||MG_2||_A \leq \left(\frac{\epsilon}{1+\epsilon}\left(1 + \left(\frac{\epsilon^{2\nu-1}}{(2+\epsilon)^{2\nu}}\right)\right)\right)^{1/2} < 1. \tag{2.9}$$

Several generalizations of both the theory and practice of reduction-based multigrid methods have since been developed. A generalization to non-symmetric M-matrices was proposed and analyzed by Mense and Nabben [19], using the tools of weak regular splitting [29]. For symmetric and positive definite problems, Brannick et al. [8] study the introduction of more general relaxation schemes, as well as the use of different approximations of A_{FF} for interpolation and relaxation. Gossler and Nabben [13] examine generalization of AMGr to the use of Chebyshev polynomial acceleration of multiple relaxation sweeps. For strongly non-symmetric systems, Manteuffel et al. [17, 18] have proposed similar approaches using so-called approximate ideal restriction (AIR) techniques, that offer excellent performance for advection-dominated problems. None of the above schemes, however, address the poor performance observed in AMGr-type methods for anisotropic problems, which is the motivation for the present work.

2.4.1 Coarsening in AMGr

Theorem 2.1 establishes existence of an interpolation operator, P, provided that the matrix, A, can be partitioned into $\begin{bmatrix} A_{FF} & -A_{FC} \\ -A_{FC}^T & A_{CC} \end{bmatrix}$ and an approximation, D_{FF}, of A_{FF} can be made that satisfies the assumptions in the theorem. While this is insightful, it does not address the fundamental question of how to generate a partitioning for which the assumptions hold with small parameter ϵ. To answer this question, MacLachlan and Saad [16] propose to partition the rows and columns of A in order to ensure the diagonal dominance of A_{FF}, allowing D_{FF} to be chosen as a diagonal matrix. In particular, for each row, i, the diagonal dominance of row i over the F points is

quantified by

$$\eta_i = \frac{|A_{ii}|}{\sum_{j \in F} |A_{ij}|},$$

Then, A_{FF} is said to be η-diagonally dominant if $\eta_i \geq \eta$ for all $i \in F$, for some $\eta > 1/2$ that measures the diagonal dominance of A_{FF}. If A_{FF} is η-diagonally dominant, then the diagonal matrix, D_{FF}, with $(D_{FF})_{ii} = (2 - \frac{1}{\eta})A_{ii}$ for all $i \in F$ yields $0 \preceq \mathcal{E} \preceq \frac{2-2\eta}{2\eta-1} D_{FF}$, giving an η-dependent convergence bound if the other assumptions of 2.1 are satisfied. Furthermore, if A is symmetric, positive-definite, and diagonally dominant, then this condition guarantees that all conditions of 2.1 are satisfied.

In addition to establishing this connection between the diagonal dominance parameter η and the convergence parameter, ϵ, MacLachlan and Saad [16] pose the partitioning algorithm as an optimization problem: for a given $\eta > 1/2$, find the largest F-set such that $\eta_i \geq \eta$ for every $i \in F$. This can be written as

$$\begin{aligned} &\max_{F \subset \Omega} |F|, \\ &\text{subject to } |A_{ii}| \geq \eta \sum_{j \in F} |A_{ij}|, \ \forall i \in F. \end{aligned} \tag{2.10}$$

MacLachlan and Saad [16] show that finding the optimal solution to (2.10) is NP-complete and, consequently, propose a greedy algorithm to approximately solve the optimization problem. The greedy algorithm acts iteratively, adding points to the C-set one at a time, and moving any points that are guaranteed to satisfy the inequality constraint in (2.10) into the F-set, until a full partition is computed.

Bibliography

[1] Nathan Bell, Steven Dalton, and Luke N Olson. Exposing fine-grained parallelism in algebraic multigrid methods. *SIAM Journal on Scientific Computing*, 34(4): C123–C152, 2012.

[2] WN Bell. *Algebraic multigrid for discrete differential forms*. Ph.D. thesis, University of Illinois at Urbana-Champaign, 2008.

[3] Dietrich Braess. Towards algebraic multigrid for elliptic problems of second order. *Computing*, 55(4):379–393, 1995.

[4] A. Brandt, S. F. McCormick, and J. W. Ruge. Algebraic multigrid (AMG) for automatic multigrid solutions with application to geodetic computations. Technical report, Inst. for Computational Studies, Fort Collins, Colo., 1982.

[5] A. Brandt, S. F. McCormick, and J. W. Ruge. Algebraic multigrid (AMG) for sparse matrix equations. In D. J. Evans, editor, *Sparsity and Its Applications*, pages 257–284. Cambridge University Press, Cambridge, 1984.

[6] Achi Brandt, James Brannick, Karsten Kahl, and Ira Livshits. An algebraic distances measure of AMG strength of connection. *arXiv preprint arXiv:1106.5990*, 2011.

[7] James Brannick, Marian Brezina, S MacLachlan, T Manteuffel, S McCormick, and J Ruge. An energy-based amg coarsening strategy. *Numerical Linear Algebra with Applications*, 13(2-3):133–148, 2006.

[8] James Brannick, A Frommer, Karsten Kahl, Scott MacLachlan, and Ludmil Zikatanov. Adaptive reduction-based multigrid for nearly singular and highly disordered physical systems. *Electronic Transactions on Numerical Analysis*, 37: 276–295, 2010.

[9] Marian Brezina, R Falgout, Scott MacLachlan, T Manteuffel, S McCormick, and J Ruge. Adaptive smoothed aggregation (αSA) multigrid. *SIAM Review*, 47(2): 317–346, 2005.

[10] William L Briggs, Van Emden Henson, and Steve F McCormick. *A multigrid tutorial*. SIAM, 2000.

[11] Jeff Erickson. Algorithms, 2019. URL `http://jeffe.cs.illinois.edu/teaching/algorithms/`.

[12] R.D. Falgout and P.S. Vassilevski. On generalizing the AMG framework. *SIAM J. Numer. Anal.*, 42(4):1669–1693, 2004.

[13] Florian Gossler and Reinhard Nabben. On AMG methods with F-smoothing based on Chebyshev polynomials and their relation to AMGr. *Electron. Trans. Numer. Anal.*, 45:146–159, 2016.

[14] Stuart Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[15] S. MacLachlan, T. Manteuffel, and S. McCormick. Adaptive reduction-based AMG. *Numer. Linear Alg. Appl.*, 13:599–620, 2006.

[16] Scott MacLachlan and Yousef Saad. A greedy strategy for coarse-grid selection. *SIAM Journal on Scientific Computing*, 29(5):1825–1853, 2007. doi: 10.1137/060654062.

[17] Thomas A Manteuffel, John Ruge, and Ben S Southworth. Nonsymmetric algebraic multigrid based on local approximate ideal restriction (LAIR). *SIAM Journal on Scientific Computing*, 40(6):A4105–A4130, 2018.

[18] Thomas A Manteuffel, Steffen Münzenmaier, John Ruge, and Ben Southworth. Nonsymmetric reduction-based algebraic multigrid. *SIAM Journal on Scientific Computing*, 41(5):S242–S268, 2019.

[19] Christian Mense and Reinhard Nabben. On algebraic multilevel methods for nonsymmetric systems-convergence results. *Electronic Transactions on Numerical Analysis*, 30:323–345, 2008.

[20] Stanislav Míka and Petr Vaněk. Acceleration of convergence of a two-level algebraic algorithm by aggregation in smoothing process. *Applications of Mathematics*, 37(5):343–356, 1992.

[21] Luke N Olson, Jacob Schroder, and Raymond S Tuminaro. A new perspective on strength measures in algebraic multigrid. *Numerical Linear Algebra with Applications*, 17(4):713–733, 2010.

[22] M. Ries, U. Trottenberg, and G. Winter. A note on MGR methods. *J. Lin. Alg. Applic.*, 49:1–26, 1983.

[23] J. W. Ruge and K. Stüben. Algebraic multigrid (AMG). In S. F. McCormick, editor, *Multigrid Methods*, volume 3 of *Frontiers in Applied Mathematics*, pages 73–130. SIAM, Philadelphia, PA, 1987.

[24] K. Stüben. Algebraic multigrid (AMG): An introduction with applications. *Multigrid*, 2000.

[25] Paul N. Swarztrauber. The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle. *SIAM Rev.*, 19(3):490–501, 1977. ISSN 0036-1445. doi: 10.1137/1019071.

[26] P Vaněk, J Mandel, and M Brezina. Algebraic multigrid based on smoothed aggregation for second and fourth order problems. *Computing*, 56:179–196, 1996.

[27] P Vaněk, M Brezina, and J Mandel. Convergence of algebraic multigrid based on smoothed aggregation. *Numerische Mathematik*, 88(3):559–579, 2001.

[28] Petr Vaněk. Acceleration of convergence of a two-level algorithm by smoothing transfer operators. *Applications of Mathematics*, 37(4):265–274, 1992.

[29] R. S. Varga. *Matrix Iterative Analysis*. Springer Series in Computational Mathematics. Springer, Berlin, 2000. Second Edition.

Chapter 3

Coarse-Grid Selection Using Simulated Annealing

Abstract

[1] Multilevel techniques are efficient approaches for solving the large linear systems that arise from discretized partial differential equations and other problems. While geometric multigrid requires detailed knowledge about the underlying problem and its discretization, algebraic multigrid aims to be less intrusive, requiring less knowledge about the origin of the linear system. A key step in algebraic multigrid is the choice of the coarse/fine partitioning, aiming to balance the convergence of the iteration with its cost. In work by MacLachlan and Saad [29], a constrained combinatorial optimization problem is used to define the "best" coarse grid within the setting of a two-level reduction-based algebraic multigrid method and is shown to be NP-complete. Here, we develop a new coarsening algorithm based on simulated annealing to approximate solutions to this problem, which yields improved results over the greedy algorithm developed previously. We present numerical results for test problems on both structured and unstructured meshes, demonstrating the ability to exploit knowledge about the underlying grid structure if it is available.

Keyword: Algebraic Multigrid, Coarse-Grid Selection, Simulated Annealing,

[1] This work is under revision as "Coarse-Grid Selection Using Simulated Annealing", by T. U. Zaman, S. P. MacLachlan, L. N. Olson, and M. West, for Journal of Computational and Applied Mathematics, 2023.

Combinatorial Optimization.

3.1 Introduction

Multigrid and other multilevel methods are well-established algorithms for the solution of large linear systems of equations that arise in many areas of computational science and engineering. Multigrid methods arise from the observation that basic iterative methods, such as the (weighted) Jacobi and Gauss-Seidel iterations, effectively eliminate only part of the error in an approximation to the solution of the problem, and that the complementary space of errors can be corrected from a coarse representation of the problem. While geometric multigrid has been shown to be highly effective for problems where a hierarchy of discrete models can be built directly, many problems benefit from the use of algebraic multigrid (AMG) techniques, where graph-based algorithms and other heuristics are used to define the multigrid hierarchy directly from the matrix of the finest-grid problem.

First introduced in the early 1980's [5, 6], AMG is now widely used and available in standard libraries, such as hypre [16, 21], PETSc [2, 3], Trilinos [18, 22], and PyAMG [32]. While there are many variants of AMG (as discussed below), the common features of AMG algorithms are the use of graph-based algorithms to construct a coarse grid from the given fine-grid discretization matrix (possibly with some additional information, such as geometric locations of the degrees of freedom for elasticity problems [40]) followed by construction of an algebraic interpolation operator from the coarse grid to the fine grid. Both of these components have received significant attention in the literature, with an abundance of schemes for creating the coarse grid and for determining the entries in interpolation. In this paper, we focus on the coarse-grid correction process, adopting a combinatorial optimization viewpoint on the selection of coarse-grid points in the classical AMG setting.

One of the primary differences between so-called classical (Ruge-Stüben) AMG and (smoothed) aggregation approaches is in how the coarse grid itself is defined [37]. In aggregation-based approaches, coarse grids are defined by clustering fine-grid points into aggregates (or subdomains), while, in classical AMG, a subset of the fine-grid degrees of freedom (points) is selected in order to form the coarse grid. In the original method [6, 35], this was accomplished using a greedy algorithm to construct a maximal independent set over the fine-grid degrees of freedom as a "first pass" at forming the coarse-grid set, which is then augmented by a "second pass" algorithm

that ensures suitable interpolation can be defined from the final coarse grid. Many alternative strategies to forming the coarse grid have been proposed over the years, particularly for the parallel case, to overcome the inherent sequentiality of the original algorithm [1, 12, 13, 14, 21]. Other approaches, for example based on redefining the notion of strength of connection [7, 10, 11, 33] or use of compatible relaxation principles [9, 24], have also been proposed.

Much of this work has been motivated by the failure of the classical AMG heuristics to yield robust solvers for difficult classes of problems, such as the Helmholtz equation, convection-dominated flows, or coupled systems of PDEs. While much research has tried to generalize these heuristics to give sensible algorithms for broader classes of problems, another approach is to consider methods that abandon these heuristics in favour of algorithms that are directly tied to rigorous multigrid convergence theory. In recent years, two main classes of AMG algorithms have arisen in this direction. One class of methods arises in the aggregation setting, where well-chosen pairwise aggregation algorithms coupled with suitable relaxation can yield convergence guarantees for symmetric, diagonally dominant M-matrices [8, 31]. The second class of methods are those based on reduction-based AMG principles, where there is a direct connection between properties of the fine-grid submatrix and the guaranteed convergence rate of the scheme [19, 27, 28, 29]. Reduction-based multigrid methods [34] are a generalization of cyclic reduction [38], in which conditions on the coarse/fine partitioning are relaxed so that while the exact Schur complement may not be sparse, it can be accurately approximated by a sparse matrix. This notion is made rigorous by MacLachlan et al. [28], who propose a theoretical framework to analyze convergence based on diagonal dominance of the fine-grid submatrix (in contrast to the fine-grid submatrix being diagonal in the classical setting of cyclic reduction), and extended to Chebyshev-style relaxation schemes by Gossler and Nabben [19]. While the theoretical guarantees offered by such methods are attractive, these approaches suffer from two key limitations. First, the classes of problems for which convergence rates are guaranteed are limited (requiring diagonal dominance and/or M-matrix structure). Second, the guaranteed convergence rates of stationary iterations are generally poor in comparison to methods based on classical heuristics. An active area of research (disjoint from the goals of this paper) is addressing the first limitation [30], while the second can be ameliorated by the use of Krylov subspace methods to accelerate stationary convergence. We note that there are other families of theoretical analysis of AMG algorithms [26], including theory tailored to heuristic approaches, such as

compatible relaxation [9]. Much of this theory, however, is of a confirmational and not a predictive nature, i.e., the convergence bounds rely on properties of the output of heuristics for choosing coarse grids and interpolation operators, and not on the inputs to these processes, such as the system matrix and algorithmic parameters. Thus, while it provides guidance in the development of heuristic methods, it does not provide the concrete convergence bounds offered by pairwise-aggregation or AMGr methods.

While the pairwise aggregation methodology [8, 31] provides practical algorithms to generate coarse grids that satisfy their convergence guarantees, this is a notable omission in the work of MacLachlan et al. [28] and much of the work on AMGr. MacLachlan and Saad [27, 29] identify that the choice of optimal coarse grids for AMGr can be quantified as the solution of an NP-complete integer linear programming problem. They use this formulation to guide construction of a family of greedy coarsening algorithms that aim to maximize the size of the fine-grid set subject to maintaining a fixed level of diagonal dominance in the fine-grid submatrix (and, consequently, the resulting convergence bound on the AMG method). While the greedy algorithm was tested on a range of problems in [29], these problems were limited primarily to bilinear finite-element discretization on structured grids, with only a few matrices outside this class. When we assessed its performance on a broader class of problems, we exposed some simple test cases where it dramatically fails to perform well, such as standard five-point finite difference discretization, motivating further work in this area. In this paper, we follow the theory and framework for AMGr [28, 29], but aim to overcome some limitations of the underlying greedy coarsening algorithm. In particular, we develop a new coarsening algorithm based on simulated annealing to partition the coarse and fine points. Numerical results show that this algorithm achieves smaller coarse grids (than the greedy approach) that satisfy the same diagonal-dominance criterion. Hence, these grids lead to moderately more efficient (two-level) algorithms, by reducing the size of the coarse-grid problem without changing the convergence bound guaranteed by the theory. We emphasize that this work is presented more as a proof-of-concept than as a practical coarsening algorithm, due to the high cost of simulated annealing. In related work, we have shown that the "online" cost of the approach proposed here can be traded for a significant "offline" cost using a reinforcement learning approach [39]. However, a necessary next step in the work proposed here is in investigating more cost-effective alternatives to these methods.

This paper is arranged as follows. In Section 3.2, AMG coarsening and the existing greedy coarsening algorithm for AMGr are discussed. The new coarsening algorithm based on simulated annealing is outlined in Section 3.3. Numerical experiments are given in Section 3.4, demonstrating performance of this approach on both isotropic and anisotropic problems, discretized on both structured and unstructured meshes. Concluding remarks and potential future research are discussed in Section 3.5.

3.2 AMG coarsening

Geometric multigrid is known to be highly effective for many problems discretized on structured meshes. However, it is naturally more difficult to make effective coarse grids (and effective coarse-grid operators) for problems discretized on unstructured meshes. AMG was developed specifically to address this, automating the formation of coarse-grid matrices without any direct mesh information. The first coarsening approach in AMG, often called classical or Ruge-Stüben (RS) coarsening, was developed by Brandt, McCormick, and Ruge [5, 6] and later by Ruge and Stüben [35]. We summarize this approach below, primarily to allow comparison with the reduction-based AMG method [28, 34] and the greedy coarsening approach proposed by MacLachlan and Saad [29] that is the starting point for the research reported herein.

AMG algorithms make use of a two-stage approach, where a setup phase precedes the cycling in the solve phase. Common steps in AMG setup phase include recursively choosing a coarse grid and defining intergrid transfer operators. The details of these processes depend on the specifics of the AMG algorithm under consideration, with both point-based and aggregation-based approaches to determining the coarse grid, and a variety of interpolation schemes possible for both of these.

3.2.1 Classical coarsening

As in geometric multigrid, the coarse grid in AMG is selected so that errors not reduced by relaxation can be accurately approximated on coarse grids. In an effective scheme, these errors are interpolated accurately from coarse grids that have substantially fewer degrees of freedom than the next finer grid, thus significantly reducing the cost of solving the coarse-grid residual problem.

In classical AMG, for an $n \times n$ matrix A, the index set $\Omega = \{1, \ldots, n\}$ is split into sets C and F, with $\Omega = C \cup F$ and $C \cap F = \emptyset$. Each degree of freedom (DoF) or point

$i \in C$ is denoted a C-point and a point $j \in F$ is denoted an F-point. This splitting is constructed by considering the so-called *strong* connections in the graph of matrix between the fine-level variables. Given a threshold value, $0 < \theta \leq 1$, the variable u_j is said to strongly influence u_i if $-A_{ij} \geq \theta \max_{k \neq i}\{-A_{ik}\}$, where A_{ij} is the coefficient of u_j in the ith equation. The set of points that strongly influence i, denoted by S_i, is defined as the set of points on which point i strongly depends. The set of points that are strongly influenced by i is denoted by S_i^T. Two heuristics are followed in RS coarsening to select a coarse grid:

H1 : For every F-point, i, every point $j \in S_i$ should either be a coarse-grid point or should strongly depend on at least one point in C that also strongly influences i.

H2 : The set of C-points should be a maximal subset of all points, where no C-point strongly depends on another C-point.

In practice, strong enforcement of both of *H1* and *H2* is not always possible; the classical interpolation formula relies on *H1* being strongly enforced, while *H2* is used only to encourage the selection of small, sparse coarse grids.

3.2.2 Greedy coarsening and underlying optimization

While the classical coarsening algorithm has proven effective for many problems when coupled with suitable construction of the interpolation operator [35], the process provides few guarantees in practice. Indeed, there is an inherent disconnect between the selection of any single coarse-grid point and the impact on the quality of the resulting interpolation operator. This has motivated coupled approaches to selecting coarse grids and defining interpolation, including *reduction-based* AMG, or AMGr.

Reduction-based multigrid was proposed by Ries et al. [34], building on earlier work aiming to improve multigrid convergence for the standard finite-difference (FD) Poisson problem. The fundamental idea of reduction-based multigrid lies in defining the multigrid components to approximate those of cyclic-reduction algorithms [38]. In particular, one way to interpret cyclic reduction is as a two-level multigrid method with idealized relaxation, interpolation, and restriction operators. Suppose that the coarse/fine partitioning is already determined, and consider the reordering of the

linear system $A\boldsymbol{x} = \boldsymbol{b}$ according to the partition, writing

$$A = \begin{bmatrix} A_{FF} & -A_{FC} \\ -A_{CF} & A_{CC} \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_F \\ \boldsymbol{x}_C \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}_F \\ \boldsymbol{b}_C \end{bmatrix}.$$

An *exact* algorithm for the solution of $A\boldsymbol{x} = \boldsymbol{b}$ in this partitioned form is given by

1. $\boldsymbol{y}_F = A_{FF}^{-1}\boldsymbol{b}_F$,
2. Solve $\left(A_{CC} - A_{CF}A_{FF}^{-1}A_{FC}\right)\boldsymbol{x}_C = \boldsymbol{b}_C + A_{CF}\boldsymbol{y}_F$,
3. $\boldsymbol{x}_F = \boldsymbol{y}_F + A_{FF}^{-1}A_{FC}\boldsymbol{x}_C$.

This can be turned into an iterative method for solving $A\boldsymbol{x} = \boldsymbol{b}$ in the usual way, by replacing the right-hand side vector, $\boldsymbol{b}$, by the evolving residual, and computing updates to a current approximation, $\boldsymbol{x}^{(k)}$, giving

1. $\boldsymbol{x}_F^{(k+1/2)} = \boldsymbol{x}_F^{(k)} + A_{FF}^{-1}\left(\boldsymbol{b}_F - A_{FF}\boldsymbol{x}_F^{(k)} + A_{FC}\boldsymbol{x}_C^{(k)}\right)$,
2. Solve $\left(A_{CC} - A_{CF}A_{FF}^{-1}A_{FC}\right)\boldsymbol{y}_C = \boldsymbol{b}_C + A_{CF}\boldsymbol{x}_F^{(k+1/2)} - A_{CC}\boldsymbol{x}_C^{(k)}$,
3. $\boldsymbol{x}_C^{(k+1)} = \boldsymbol{x}_C^{(k)} + \boldsymbol{y}_C$,
4. $\boldsymbol{x}_F^{(k+1)} = \boldsymbol{x}_F^{(k+1/2)} + A_{FF}^{-1}A_{FC}\boldsymbol{y}_C$.

In this form, this remains an exact algorithm: given any initial guess, $\boldsymbol{x}^{(0)}$, we have the solution $\boldsymbol{x} = \boldsymbol{x}^{(1)}$. A truly iterative method results from approximating the three instances of A_{FF}^{-1} in the above algorithm, namely

$$\hat{A}_{FF}^{-1} \approx A_{FF}^{-1} \qquad \hat{A}_C \approx A_{CC} - A_{CF}A_{FF}^{-1}A_{FC} \qquad W_{FC} \approx A_{FF}^{-1}A_{FC}, \tag{3.1}$$

leading to the following iteration:

1. $\boldsymbol{x}_F^{(k+1/2)} = \boldsymbol{x}_F^{(k)} + \hat{A}_{FF}^{-1}\left(\boldsymbol{b}_F - A_{FF}\boldsymbol{x}_F^{(k)} + A_{FC}\boldsymbol{x}_C^{(k)}\right)$,
2. Solve $\hat{A}_C\boldsymbol{y}_C = \boldsymbol{b}_C + A_{CF}\boldsymbol{x}_F^{(k+1/2)} - A_{CC}\boldsymbol{x}_C^{(k)}$,
3. $\boldsymbol{x}_C^{(k+1)} = \boldsymbol{x}_C^{(k)} + \boldsymbol{y}_C$,
4. $\boldsymbol{x}_F^{(k+1)} = \boldsymbol{x}_F^{(k+1/2)} + W_{FC}\boldsymbol{y}_C$.

In multigrid terminology, the first step can be seen as a so-called F-relaxation step, where we approximate A_{FF}^{-1} using some simple approximation, such as with weighted Jacobi or Gauss-Seidel. The second step is a coarse-grid correction step, where the residual after F-relaxation is restricted by injection, and the coarse-grid correction, $\boldsymbol{y}_C$, is computed by solving a linear system with an approximation, $\hat{A}_C$, of the true Schur complement, $A_{CC} - A_{CF}A_{FF}^{-1}A_{FC}$. The final two steps correspond to an interpolation of the coarse-grid correction, writing the interpolation operator $P = \left[\begin{smallmatrix} W_{FC} \\ I \end{smallmatrix}\right]$, for some approximation of the ideal interpolation operator, $W_{FC} \approx A_{FF}^{-1}A_{FC}$. If the approximation in the first step is sufficiently poor that a significant residual is expected to remain at the F-points after relaxation, it is also reasonable to augment the restriction in the second step, using either the transpose of interpolation or some better approximation to an ideal restriction operator than just injection.

As written above, the algebraic form of reduction-based multigrid retains the key disadvantage of classical AMG — there is little connection between the algorithmic choices (of the coarse/fine partitioning, and the approximations of A_{FF}, of the Schur complement, and of the idealized interpolation and restriction operators) with the resulting convergence of the algorithm. To address these limitations, MacLachlan et al. [28] proposed a reduction-based AMG algorithm (AMGr) that attempts to connect convergence with properties of A_{FF}. In particular, they presented a two-level convergence theory with a convergence rate quantified by the approximation of $A_{FF} \approx D_{FF}$, with the expectation that application of D_{FF}^{-1} to both a vector and to A_{FC} is computationally feasible. In what follows, we use the standard notation that matrices $A \succeq B$ when $\boldsymbol{x}^T A\boldsymbol{x} \geq \boldsymbol{x}^T B\boldsymbol{x}$ for all vectors $\boldsymbol{x}$.

Theorem 3.1. *[28] Consider the symmetric and positive-definite matrix* $A = \left[\begin{smallmatrix} A_{FF} & -A_{FC} \\ -A_{FC}^T & A_{CC} \end{smallmatrix}\right]$ *such that* $A_{FF} = D_{FF} + \mathcal{E}$*, with* D_{FF} *symmetric,* $0 \preceq \mathcal{E} \preceq \epsilon D_{FF}$*, and* $\left[\begin{smallmatrix} D_{FF} & -A_{FC} \\ -A_{FC}^T & A_{CC} \end{smallmatrix}\right] \succeq 0$*, for some* $\epsilon \geq 0$*. Define relaxation with error-propagation operator* $R = \left(I - \sigma\left(\begin{smallmatrix} D_{FF}^{-1} & 0 \\ 0 & 0 \end{smallmatrix}\right)A\right)$ *for* $\sigma = 2/(2+\epsilon)$*, interpolation* $P = \left[\begin{smallmatrix} D_{FF}^{-1}A_{FC} \\ I \end{smallmatrix}\right]$*, and coarse-level correction with error-propagation operator* $T = I - P(P^TAP)^{-1}P^TA$*. Then the multigrid cycle with* ν *pre-relaxation sweeps, coarse-level correction, and* ν *post-relaxation sweeps has error propagation operator* $MG_2 = R^\nu \cdot T \cdot R^\nu$ *which satisfies*

$$||MG_2||_A \leq \left(\frac{\epsilon}{1+\epsilon}\left(1 + \left(\frac{\epsilon^{2\nu-1}}{(2+\epsilon)^{2\nu}}\right)\right)\right)^{1/2} < 1. \tag{3.2}$$

If a partitioning and approximation, D_{FF}, of A_{FF} are found that satisfy the assumptions given above, then this theorem establishes existence of an interpolation

operator, P, giving multigrid convergence with a direct tie to the approximation parameter, ϵ. In this theory, tight spectral equivalence between D_{FF} and A_{FF} is required to ensure good performance of the solver. This leads to the goal of constructing the fine-grid set so that there is a guarantee of tight spectral equivalence between D_{FF} and A_{FF}.

To meet that goal, MacLachlan and Saad [29] proposed to partition the rows and columns of the matrix A in order to ensure the diagonal dominance of A_{FF}, so that D_{FF} could be chosen as a diagonal matrix. In particular, a diagonal dominance factor is introduced for each row i, defined as

$$\theta_i = \frac{|A_{ii}|}{\sum_{j\in F} |A_{ij}|},$$

With this, A_{FF} is said to be θ-diagonally dominant if $\theta_i \geq \theta$ for all $i \in F$, where $\theta > 1/2$ measures the diagonal dominance of A_{FF}. If A_{FF} is θ-diagonally dominant, then it is shown that the diagonal matrix, D_{FF}, with $(D_{FF})_{ii} = (2 - \frac{1}{\theta})a_{ii}$ for all $i \in F$ leads to $0 \preceq \mathcal{E} \preceq \frac{2-2\theta}{2\theta-1} D_{FF}$, giving a θ-dependent convergence bound if the other assumptions of Theorem 3.1 are satisfied. Furthermore, if A is symmetric, positive-definite, and diagonally dominant, then this prescription for D_{FF} guarantees that all conditions of Theorem 3.1 are satisfied, so long as A_{FF} is θ-diagonally dominant.

In addition to establishing this connection between the diagonal dominance parameter θ and the convergence parameter, ϵ, MacLachlan and Saad [29] posed the partitioning algorithm as an optimization problem for given $\theta > 1/2$, asking for the largest F-set such that $\theta_i \geq \theta$ for every $i \in F$. Such a θ-dominant A_{FF} guarantees good equivalence between a diagonal matrix, D_{FF}, and A_{FF}, and the largest such F-set would make the coarse-grid problem smallest. This leads to an optimization problem of the form

$$\begin{aligned} &\max_{F\subset\Omega} |F|, \\ &\text{subject to } |A_{ii}| \geq \theta \sum_{j\in F} |A_{ij}|, \ \forall i \in F. \end{aligned} \tag{3.3}$$

The greedy algorithm [29] for (3.3) selects rows and columns of the matrix A for A_{FF} in a greedy manner to ensure the diagonal dominance of A_{FF}, as described in Algorithm 3.1. Here, the set U contains all the DoFs that are unpartitioned (not yet assigned to be coarse or fine points). Initially, all degrees of freedom are assigned to U, while the F and C sets are empty. The rows that directly satisfy the diagonal dominance criterion are initially added to the F-set (and removed from U).

If there are no diagonally dominant DoFs in the U set, then the point having the least diagonal dominance is made a C-point. This selection may make some other points in U diagonally dominant, whereupon these DoFs are added to the F-set and removed from U, and the process is repeated until the set U becomes empty.

Algorithm 3.1 Greedy coarsening algorithm

function GREEDY-COARSENING(A, θ)
  $U \leftarrow \{1, 2, \ldots, n\}$, $F \leftarrow \emptyset$, $C \leftarrow \emptyset$
  for $i \leftarrow 1, \ldots, n$ **do**
    $\hat{\theta}_i \leftarrow \frac{|A_{ii}|}{\sum_{j \in F \cup U} |A_{ij}|}$
    if $\hat{\theta}_i \geq \theta$ **then**
      $F \leftarrow F \cup \{i\}$, $U \leftarrow U \setminus \{i\}$
  while $U \neq \emptyset$ **do**
    $j \leftarrow \text{argmin}_{i \in U} \{\hat{\theta}_i\}$
    $U \leftarrow U \setminus \{j\}$, $C \leftarrow C \cup \{j\}$
    for $i \in U \cap \text{Adj}(j)$ **do** ▷ $\text{Adj}(j) = \{k : A_{jk} \neq 0\}$
      $\hat{\theta}_i \leftarrow \frac{|A_{ii}|}{\sum_{k \in F \cup U} |A_{ik}|}$
      if $\hat{\theta}_i \geq \theta$ **then**
        $F \leftarrow F \cup \{i\}$, $U \leftarrow U \setminus \{i\}$
  Return F, C

We note that there is a slightly counter-intuitive connection between the quality of solution to (3.3) and the convergence of the multigrid method. "Bad" solutions to (3.3), meaning those with satisfied constraints but that are far from optimality, have much bigger C-sets than "good" solutions do and, consequently, the resulting two-level convergence can be much better than that guaranteed by the theoretical bounds. Extreme examples of this occur when the set F is an independent set, so that A_{FF} is a diagonal matrix, and the resulting two-grid method is exact. While such cases can be treated by the analysis in [29], they rely on *a posteriori* measurement of the largest θ for which the constraints in (3.3) hold, rather than the *a priori* bound that is given by the initial choice of the value of θ for which we try to solve (3.3). Thus, our goal in optimizing (3.3) is to improve complexities of the resulting algorithm (by making the F-set larger), subject to the same two-level convergence bound fixed a priori by our choice of θ. We note that this is further complicated by the complex

relationship between two-level and multilevel convergence and complexities. In particular, when coarsening rates are slow (corresponding to "bad" solutions of (3.3)), two-level convergence is generally a bad predictor of multilevel convergence (since good two-level convergence relies on an assumption of exact solution of large coarse-grid matrices). We present the supporting numerical results to explore this connection in Section 3.4.2, but note that the work proposed here seeks specifically to improve the quality of solution to (3.3), by finding larger F-sets that satisfy the constraints for a specified value of θ.

While the convergence bound for AMGr is attractive, it has several shortcomings. First and foremost is the strict assumption on diagonal dominance needed for the convergence guarantee to be valid — we explore cases where this is not the case below in Sections 3.4.3 and 3.4.4, and see that these problems are, indeed, more difficult to handle in this framework. Second, we emphasize that while the convergence bound obtained in Theorem 3.1 depends only on ϵ (and, thus, is independent of mesh size if the coarse grid is generated by Algorithm 3.1 or the techniques introduced below with constant θ). These bounds are generally worse than the observed convergence for standard AMG approaches. Using $\theta = 0.56$, as we do for all examples below, gives $\epsilon = 22/3$, and the convergence bound for $\nu = 1$ in (3.2) is 0.977. While this can be improved by using either more relaxation per cycle or by accelerating convergence with a Krylov method, we consider only stationary cycles here, accepting poorer convergence factors in exchange for a direct relationship to the convergence theory summarized above. We note that this is also similar to the stationary convergence bound for the method in Napov and Notay [31], which is 0.93 for problems that satisfy the assumptions therein.

3.3 Simulated annealing

The optimization problem in (3.3) is a combinatorial optimization problem. Because such problems arise in many areas of computational science and engineering, significant research effort has been devoted to developing algorithms for their solution, both of general-purpose type (e.g., branch and bound techniques) and for specific problems (e.g., the travelling salesman problem) [23]. For many problems, the size of the solution space makes exhaustive brute-force algorithms infeasible; for some such problems, branch and bound techniques may be successful in paring down the solution space to a more manageable size. In many cases, however, there are no feasible

exact algorithms, and stochastic search algorithms, such as simulated annealing (SA), genetic algorithms, or tabu search methods, can be employed [36]. In this work, we apply SA algorithms to approximate the global minimum of the optimization problem in (3.3).

Simulated annealing (SA) is a probabilistic method used to find global optima of cost functions that might have a large number of local optima. The SA algorithm randomly generates a state at each iteration and the cost function is computed for that state. The value of the cost function for a state determines whether the state is an improvement. If the current state improves the value of the objective function, it is accepted to exploit the improved result. The current state might also be accepted, with some probability less than one, even if it is worse than the previous state, however the probability of accepting a bad state decreases exponentially with the "badness" of the state. The purpose of (sometimes) accepting inferior states, known as "exploration", is to avoid being trapped in local optima, and the inferior intermediate states are considered in order to give a pathway to a globally better solution. The total number of iterations of SA depends on an initial "temperature" and the rate of decrease of that temperature. The temperature also affects the probability of accepting a bad state, with the exploration phase of the algorithm becoming less probable as the temperature decreases, to ensure convergence to a global optimum.

3.3.1 Idea and generic algorithm

Consider the set F of fine points to be the current state. To find F that maximizes a fitness function $z(F)$, such as $z(F) = |F|$ in (3.3), simulated annealing proceeds as shown in Algorithm 3.2. The temperature T starts at an initial value and decays by a factor $\alpha \in (0, 1)$ at each iteration (the "cooling schedule"). A key choice when using simulated annealing is a method for choosing a neighbor state $\widetilde{F}$ that is close to the current state F. In the case of optimizing the choice of a subset $F \subset \Omega$, a straightforward choice is to choose $\widetilde{F}$ by randomly adding or removing an element from F. Finally, the function $P(z, \tilde{z}, T)$ is the probability of accepting a move from a current state with fitness z to the new state with fitness $\tilde{z}$. The standard acceptance function is

$$P(z, \tilde{z}, T) \ = \begin{cases} 1 & \text{if } \tilde{z} > z, \\ \exp\big(-(z - \tilde{z})/T\big) & \text{otherwise.} \end{cases} \tag{3.4}$$

This probability function always accepts transitions that raise the fitness, and sometimes accepts transitions that decrease the fitness. Occasionally accepting fitness-decreasing transitions is essential to escape local maxima in the energy landscape, with the chance of such transitions being controlled by the current temperature. By analogy with the physical annealing of metals, at high temperatures we will accept almost all fitness-decreasing transitions, allowing exploration of the fitness landscape, but as the temperature cools we will accept fewer of these transitions, until we eventually become trapped near a fitness maximum.

Algorithm 3.2 Generic simulated annealing (SA) algorithm

1: Initialize F to a random state and T to an initial temperature
2: **for** n_{steps} iterations **do**
3: Randomly pick a neighbor state $\widetilde{F}$ of F
4: **if** $\text{rand}(0,1) < P(z(F), z(\widetilde{F}), T)$ **then**
5: $F \leftarrow \widetilde{F}$
6: $T \leftarrow \alpha T$
7: Return F

Much work has been devoted to the design of optimal transition probability functions [17] and cooling schedules [15] in simulated annealing algorithms. While it can be shown that simulated annealing converges to the global maximum with probability one as the cooling time approaches infinity [20], in practice the performance depends significantly on the selection of the neighbors $\widetilde{F}$ of the current state F. A common heuristic for choosing neighbors is to select states with similar fitness, which is more efficient because we are less likely to reject such transitions, although this is in tension with the desire to escape from steep local maxima.

There are also different algorithmic possibilities for the handling of constraints on the state F. Given a constraint subset of allowable states, one common method of enforcing the constraint is to pick only neighbor states $\widetilde{F}$ that are in the constraint subset. This method is appropriate if the constraint subset is connected and constraint-satisfying neighbors can always be found, and it is easy to see that this algorithm inherits all theoretical properties of the unconstrained version. An alternative method, which we will use in Section 3.3.2, is to allow constraint-violating neighbors to be selected but keep a record of the highest-fitness constraint-satisfying state visited. This method is advantageous when constraint-violating paths in state

space allow rapid transitions between (possibly disconnected) areas in the constraint subset. This algorithm also preserves the global guarantees of convergence under the condition that the global maximum is constraint satisfying.

3.3.2 Adaptation to coarse/fine partitioning

While it is possible to apply SA directly to the optimization problem in (3.3), preliminary experiments show that this is inefficient, due to the global coupling of the degrees of freedom in the optimization problem. To overcome this, we consider a domain decomposition approach to solving the optimization problem, where we first divide the discrete set of degrees of freedom into (non-overlapping) subdomains (the construction of which is considered in detail in the following subsection) and apply SA to the subdomain problems. These subdomain problems are not independent, therefore information is exchanged about the tentative partitioning on adjacent subdomains; this is accomplished in either an additive (Jacobi-like) or multiplicative (Gauss-Seidel-like) manner. For sufficiently small subdomains, however, SA can efficiently find near-optimal solutions to localized versions of the optimization in (3.3), and we focus on this process below.

SA is run on each subdomain to partition its DoFs into (local) C- and F-sets. As the suitability of this partitioning in a global sense naturally depends on decisions being made on adjacent subdomains, we must be careful to consider what happens to DoFs in the global mesh adjacent to those in the subdomain. If these subdomains already have their own tentative C/F partitions computed (as is expected to be the case on all but the first sweep through the domain), then this partitioning is considered fixed, and used to guide decisions on the current subdomain. If no tentative partitioning has yet been computed on a neighbouring subdomain, then the points in the neighbouring subdomain that are connected to some DoFs of the current subdomain are considered to be F-points, while all other DoFs in the neighbouring subdomain are considered to be C-points. These assumptions are used only for the purposes of computing the partitioning on the current subdomain, to apply hard constraints on the partitioning.

Specifically, if $\Omega = \{1, 2, \ldots, n\}$ is the global set of degrees of freedom, partitioned into s disjoint subdomains as $\Omega = \cup_{k=1}^{s} \Omega_k$, then the optimization problem to be

solved on subdomain k is given as

$$\max_{F_k \subset \Omega_k} |F_k|,$$
$$\text{subject to } |A_{ii}| \geq \theta \sum_{j \in F} |A_{ij}|, \, \forall i \in \bar{F}_k,$$

where information from other subdomains enters in the constraint, both as additional points for which the constraint must be satisfied and in the right-hand side of the constraint where we sum over $j \in F$ (and not $j \in F_k$). Here, F_k is the current set of points in Ω_k that are in the F-set, and we define $C_k = \Omega_k \setminus F_k$ to be the complementary C-set. Further, we use $\bar{F}_k$ and F to denote the sets of tentative F-points in $\overline{\Omega}_k$ and Ω, respectively, where the standard "closure" notation, $\overline{\Omega}_k$, denotes the set of points j, such that either $j \in \Omega_k$ or $A_{ij} \neq 0$ for some $i \in \Omega_k$. In the localized optimization problem above, the set $\bar{F}_k$ contains both the points in F_k and any point $j \in \overline{\Omega}_k \setminus \Omega_k$ such that either j is in the F_ℓ-set in a neighbouring subdomain, Ω_ℓ, that has a tentative partition or $j \in \Omega_\ell$ for a neighbouring subdomain, Ω_ℓ, that does not yet have a tentative partition. We note that a key part of this localization is that we make choices to optimize the partitioning on the local set, Ω_k, but consider the impacts of these choices on the global F-set, not only on the local set, F_k. Thus, when we localize to subdomain Ω_k, we consider the constraints on $\bar{F}_k$, including all points in Ω where the choice of F_k could possibly lead to a constraint violation.

Within an annealing step on a given subdomain, Ω_k, we take the current tentative set F_k as the initial guess for the partitioning, with the exception of the first step, where we take $F_k = \emptyset$ for all $i \in \Omega_k$ (to ensure we start from a configuration that satisfies the constraint). As a benchmark for the annealing process, we initialize $\overline{n}_F^{(k)}$ as the largest size of a constraint-satisfying F-set seen so far on Ω_k (taken to be zero on the first iteration), and z_k to be the number of constraint-satisfying F-points in the *current* set $\bar{F}_k$. At each annealing step on Ω_k, we swap points in and out of F_k, either increasing, decreasing, or maintaining its size, with equal probability. The basic algorithm for these swaps is given in Algorithm 3.3, where we take the sets F_k and C_k as input, along with values n_F and n_C giving the numbers of points to swap from C_k to F_k and vice-versa. We note that there are two possible ways to do this swap, either selecting the elements from F_k and C_k independently and then moving the elements, or first moving the selected elements from F_k and, then, selecting the elements from the updated set C_k to move. We follow the first way as preliminary experiments suggested that it gives slightly better results.

Algorithm 3.3 swapFC(F_k, C_k, n_F, n_C)

1: $\widetilde{F}_k \leftarrow F_k$
2: $\widetilde{C}_k \leftarrow C_k$
3: Randomly select n_C points from F_k, remove the points from $\widetilde{F}_k$, and add the points to $\widetilde{C}_k$
4: Randomly select n_F points from C_k, remove the points from $\widetilde{C}_k$, and add the points to $\widetilde{F}_k$
5: Return $\widetilde{F}_k$, $\widetilde{C}_k$

Algorithm 3.4 shows the complete annealing algorithm, with inputs given by the matrix, A, its decomposition into s subdomains, $\{\Omega_k\}_{k=1}^s$, the initial temperature, T, and its decay rate, α, as well as the number of SA steps to run for each degree of freedom in A, $n_{\text{per DoF}}$ and for each degree of freedom in each cycle, $n_{\text{per DoF per cycle}}$. In each cycle, annealing is run on every subdomain, with an ordering determined as discussed below. The main annealing step on Ω_k, as given in Algorithm 3.5, then takes the form of selecting whether to increase, decrease, or maintain the size of F_k, checking if the selected action is possible, and performing it if it is. To increase the size of F_k, we first check that C_k has sufficient entries to move. If so, we increase the size of F_k by selecting $x + y$ entries of C_k to move to F_k and y entries of F_k to move to C_k, for pre-determined values of x and y (typically $x = 1$, $y = 0$, although these values could also be drawn from a suitable distribution). If the decision is made to swap points, we check that both F_k and C_k have sufficient points to swap, then swap x points from each set into the other (typically $x = 1$). Finally, if the decision is made to decrease the size of F_k, we check that it has points to remove, then move $x + y$ points from F_k to C_k and y points from C_k to F_k (ensuring $x+y > 0$; typically $x = 1$, $y = 0$). Finally, we measure the fitness of the resulting tentative F-set and decide whether or not to accept it before decrementing the temperature by the relative factor α and the number of further annealing steps to take by one.

The fitness score of a given (tentative) partition over $\overline{\Omega}_k$ is directly calculated as the number of points in the set that satisfy the diagonal dominance criterion, as outlined in Algorithm 3.6. In the acceptance step of the algorithm, given as Algorithm 3.7, we compute the fitness score for the tentative $\bar{F}_k$ and compare it to that of the current (last accepted) set $\bar{F}_k$. If the fitness score increases or remains same, then we automatically accept the step and update F_k, C_k, and z. In this case, we additionally check if all points in the current $\bar{F}_k$-set are constraint satisfying and if

Algorithm 3.4 annealing-on-$\Omega(A, \{\Omega_k\}_{k=1}^s, T, \alpha, n_{\text{per DoF}}, n_{\text{per DoF per cycle}})$

$F \leftarrow \emptyset$, $F_k \leftarrow \emptyset$ for all $k = 1, \ldots, s$
$C_k \leftarrow \Omega_k$ for all $k = 1, \ldots, s$
$\overline{n}_F^{(k)} \leftarrow 0$, $z_k \leftarrow 0$ for all $k = 1, \ldots, s$
$n_{\text{cycle}} \leftarrow n_{\text{per DoF}}/n_{\text{per DoF per cycle}}$
for $i \leftarrow 1, \ldots, n_{\text{cycle}}$ **do**
    for $k \leftarrow 1, \ldots, s$ **do**
        $n_{\text{steps}} \leftarrow n_{\text{per DoF per cycle}} \times |\Omega_k|$
        $F_k, C_k, \overline{n}_F^{(k)}, z_k, F, T \leftarrow$ annealing-on-$\Omega_k(F_k, C_k, \overline{n}_F^{(k)}, z_k, F, T, \alpha, n_{\text{steps}})$
$C \leftarrow \Omega \setminus F$
Return F, C

Algorithm 3.5 annealing-on-$\Omega_k(F_k, C_k, \overline{n}_F^{(k)}, z_k, F, T, \alpha, n_{\text{steps}})$

for n_{steps} iterations **do**
    Randomly generate $r \in \{0, 1, 2\}$ with equal probability
    if $r = 0$ & $|C_k| \geq x + y$ **then**
        $\widetilde{F}_k$, $\widetilde{C}_k \leftarrow$ swapFC$(F_k, C_k, x + y, y)$
    if $r = 1$ & $\min(|F_k|, |C_k|) > x$ **then**
        $\widetilde{F}_k$, $\widetilde{C}_k \leftarrow$ swapFC(F_k, C_k, x, x)
    if $r = 2$ & $|F_k| \geq x + y$ **then**
        $\widetilde{F}_k$, $\widetilde{C}_k \leftarrow$ swapFC$(F_k, C_k, y, x + y)$
    Construct tentative $\bar{F}_k$, $\bar{C}_k$ from $\widetilde{F}_k$, $\widetilde{C}_k$, and F
    $F_k, C_k, \overline{n}_F^{(k)}, z, F \leftarrow$ accept$(\Omega_k, \widetilde{F}_k, \widetilde{C}_k, \bar{F}_k, \overline{n}_F^{(k)}, z, T, F)$
    $T \leftarrow \alpha T$
Return F_k, C_k, $\overline{n}_F^{(k)}$, z_k, F, T

this set increases the value of $\overline{n}_F^{(k)}$. If so, we update the value of $\overline{n}_F^{(k)}$ (and update the global F-set). If $\tilde{z} < z$, then the step is accepted with a probability that decreases with temperature and $z - \tilde{z}$, but the additional check need not be done.

3.3.3 Localization and Gauss-Seidel variants

In Section 3.4, we consider both structured-grid and unstructured-grid problems; consequently, we consider both geometric and algebraic decompositions of Ω into

Algorithm 3.6 fitness($\bar{F}_k$)

Calculate diagonal dominance for each DoF in the set $\bar{F}_k$
$\tilde{z} \leftarrow$ the number of points that meet the diagonal dominance criterion
Return $\tilde{z}$

Algorithm 3.7 accept($\Omega_k, \widetilde{F}_k, \widetilde{C}_k, \bar{F}_k, \overline{n}_F^{(k)}, z, T, F$)

$\tilde{z} \leftarrow$ fitness($\bar{F}_k$)
if $\tilde{z} \geq z$ **then**
    $z \leftarrow \tilde{z}$, $F_k \leftarrow \widetilde{F}_k$, $C_k \leftarrow \widetilde{C}_k$
    if $\tilde{z} = \left|\bar{F}_k\right|$ & $\tilde{z} \geq \overline{n}_F^{(k)}$ **then**
        $\overline{n}_F^{(k)} \leftarrow \tilde{z}$
        $F \leftarrow (F \setminus \Omega_k) \cup \widetilde{F}_k$
else
    Randomly generate $x \in [0, 1]$
    if $x < e^{(-(z-\tilde{z})/T)}$ **then**
        $z \leftarrow \tilde{z}$, $F_k \leftarrow \widetilde{F}_k$, $C_k \leftarrow \widetilde{C}_k$
Return $F_k, C_k, \overline{n}_F^{(k)}, z, F$

$\{\Omega_k\}_{k=1}^s$. An important advantage of algebraic partitioning, however, is that it can be used to generate deeper multigrid hierarchies, since geometric partitioning can only be used to generate a single coarse level (which has unstructured DoF locations and, thus, does not naturally lead to further geometric partitioning). Here, we outline both strategies.

For structured grids, we can consider geometric decomposition of the fine grid into subdomains. For problems with (eliminated) Dirichlet boundary conditions, we typically satisfy the diagonal dominance criterion already at all points adjacent to a Dirichlet boundary, so these points are taken as F-points from the beginning and not included in the decomposition into subdomains. A natural strategy is to subdivide the remaining points into square or rectangular subdomains of equal size (modulo boundary/corner cases). We consider this in Section 3.4 for both finite-difference (FD) and bilinear finite-element (FE) discretizations on uniform meshes. When the number of points (in one dimension) to be decomposed is not evenly divided by the given subdomain size (in one dimension), the right-most and bottom-most subdomains on the mesh are of smaller size, given by the remainder in that division. On

structured grids, we can consider either a lexicographical Gauss-Seidel iteration over the subdomains or a four-colored Gauss-Seidel iteration. In preliminary experiments, the four-colored Gauss-Seidel iteration outperformed the lexicographical strategy by a small margin, so we use this. While the nature of the cycling is not explicitly encoded in the loop over subdomains in Algorithm 3.4, we assume that the subdomain indexing in $\{\Omega_k\}$ is consistent with the cycling strategies discussed here.

On both structured and unstructured grids, we also consider algebraic decomposition using Lloyd aggregation to define the subdomains. Lloyd aggregation, proposed by Bell [4] and given in Algorithm 3.8, is a natural application of Lloyd's algorithm [25] to subdivide the DoFs of a matrix into well-shaped subdomains. Given a desired number of subdomains (or average size per subdomain), the unit-distance graph of the matrix is constructed and one "center" point for each subdomain is randomly selected. Each (tentative) subdomain is then selected as the set of points that are closer to the subdomain center point than to any other subdomain center, using a modified form of the Bellman-Ford algorithm (Algorithm 3.9). Then, for each subdomain, the center point is reselected (again using a modified form of the Bellman-Ford algorithm), as the current centroid of the subdomain (a DoF having maximum distance from the subdomain boundary, breaking ties arbitrarily). This process is repeated (reforming subdomains around the new centers, then reassigning centers) until the assignment to subdomains (denoted by the "membership vector", $\boldsymbol{m}$) has converged.

Algorithm 3.8 lloyd-aggregation(A, V_c)

```
repeat
    d, m ← modified-bellman-ford(A, V_c)
    B ← ∅
    for i, j such that |A_{i,j}| > 0 do
        if m_i ≠ m_j then
            B ← B ∪ {i, j}
    d, x ← modified-bellman-ford(A, B)
    V_c ← {i ∈ Ω : d_i > d_j  ∀m_i = m_j}
until no change in V_c and m
Return m
```

In the algebraic case, a multicoloured Gauss-Seidel iteration strategy is slightly more complicated, since we would need to compute a colouring of the subdomains;

Algorithm 3.9 modified-bellman-ford(A, V_c)

$d_i = \infty$ for all $i = 1, \ldots, |\Omega|$ ▷ shortest-path distance from node i to nearest center
$m_i = -1$ for all $i = 1, \ldots, |\Omega|$ ▷ cluster index (membership) containing node i
for $c \in V_c$ **do**
    $d_c \leftarrow 0$
    $m_c \leftarrow c$
while True **do**
    done $\leftarrow$ True
    for i, j such that $|A_{i,j}| > 0$ **do**
        if $d_i + d_{ij} < d_j$ **then**
            $d_j \leftarrow d_i + d_{ij}$
            $m_j \leftarrow m_i$
            done $\leftarrow$ False
    if done **then**
        Return $\boldsymbol{d}, \boldsymbol{m}$

hence, we simply use lexicographical Gauss-Seidel to iterate between subdomains, noting that the advantage of the four-color iteration in the structured case is quite small. While the results in this paper are generated using a serial implementation and lexicographical Gauss-Seidel, the algorithm could easily be parallelized using a multicoloured Gauss-Seidel iteration.

3.3.4 Benchmark results

A natural comparison is with that of the greedy strategy [29], which we provide in the following numerical results. For the structured-grid discretizations, we have also explored optimization "by hand", meaning pencil-and-paper analysis of strategies that try and maximize the size of the global F-set while satisfying the constraint.

For the five-point finite-difference stencil of the Laplacian on a uniform 2D mesh, for any value $\frac{1}{2} < \theta < \frac{4}{7}$, the diagonal dominance criterion will be satisfied if every F-point has at least one C-neighbour. An optimal strategy [39] for this case arises by dividing the fine mesh into "X-pentominos", sets of five grid points with one center point and its four cardinal neighbours, plus edge/corner cases where only a subset of an X-pentomino is needed. Then, the C-set can be selected as the center points of

each X-pentomino (or subset of one), with the remaining points assigned to the F-set. In an ideal case (e.g., with periodic boundary conditions, or minimal edge cases), this leads to an F-set with size equal to $4/5$ of the size of Ω. Such a coarsening is shown in the left panel of Figure 3.1. Note that some points adjacent to Dirichlet BCs are naturally chosen as coarse points in this strategy, despite their inherent diagonal dominance, because they are center points of an X-pentomino that includes just one point in the region away from the boundary. These could be equally well treated by making the interior point in these X-pentominos a C-point and leaving the center point on the boundary as an F-point, but there is no advantage to doing so.

For the nine-point bilinear finite-element discretization of the Laplacian on a uniform 2D mesh, for any value of θ less than $4/7$, the diagonal dominance criterion will be satisfied if every F-point has at least two C-neighbours. Here, we partition the points (again except those adjacent to a Dirichlet boundary) into square subdomains of size 3×3, and select two points consistently in each subdomain as C-points. In an ideal case, where we can fill the domain with 3×3 "bricks", this leads to an F-set with $7/9$ of the size of Ω. Such a coarsening is shown in the right panel of Figure 3.1. When the domain cannot be filled perfectly with 3×3 bricks, the remaining square/rectangular regions still require one or two C points to be selected, reducing the optimal size of the resulting F set.

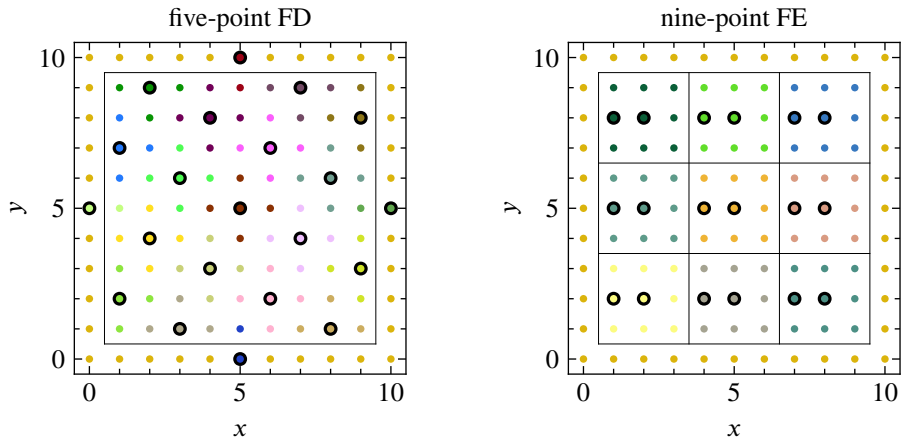


Figure 3.1: Splitting of C- and F- points for 11×11 meshes from optimization "by hand". At left, X-pentomino coarsening for five-point FD scheme. At right, 3×3 brick coarsening for nine-point FE scheme. At left, we color by the X-pentominos and at right, we color by the 3×3 bricks. C-points are represented by the black circles.

3.4 Results

In the results below, unless stated otherwise, we set the initial (global) temperature at $T = 1$, and compute the reduction rate, α, so that $\alpha^{n_{\text{ts}}} = 0.1$, where n_{ts} is the total number of annealing steps to be attempted. Thus, these experiments end when the global temperature reaches 0.1. In addition, for each problem, we determine n_{ts} by fixing a total number of steps per DoF in the system, and report results based on the allotted number of steps per DoF, $n_{\text{per DoF}}$. This work is further subdivided into Gauss-Seidel sweeps by fixing values of the number of annealing steps per DoF per sweep, $n_{\text{per DoF per cycle}}$, with the number of sweeps determined as the ratio of total steps per DoF to steps per DoF per sweep, as on Line 4 of Algorithm 3.4.

We first consider structured-grid experiments for both the finite-difference (five-point) and bilinear finite element (nine-point) discretizations of the Laplacian, using a 32×32 mesh to explore how much work is needed to determine near-optimal partitions using both geometric and algebraic divisions into subdomains. Using these experiments to identify "best practices", we then explore how the coarsening algorithm performs as we change the problem size, move from structured to unstructured finite-element meshes, and isotropic to anisotropic operators.

3.4.1 Structured-grid discretizations with geometrically structured subdomains

We start by considering the five-point finite-difference stencil on a fixed (uniform) 32×32 mesh, and consider the effects of changing both the size of the Gauss-Seidel subdomains and the distribution of work in the algorithm. As a measure of quality of the results, we consider the maximum of the ratio $|F|/|\Omega|$ over all constraint-satisfying F-sets generated in a single run of the annealing algorithm. As a comparison, for this problem, the best optimization "by hand" of the size of the F-set yields $|F|/|\Omega| = 0.8047$, as depicted by the black lines in Figure 3.2, while the greedy algorithm [29] yields $|F|/|\Omega| = 0.561$ (not depicted because it is far from the data shown here). We vary three algorithmic parameters in Figure 3.2, the total number of SA steps per DoF, with values ranging from 5000 to 2 000 000, the number of SA steps per DoF in a single sweep of Gauss-Seidel on each subdomain, and the size of the subdomains used in the Gauss-Seidel sweeps.

We can draw three conclusions from the data presented in Figure 3.2. First, we note that if the subdomains are "too small", then there is little benefit in investing

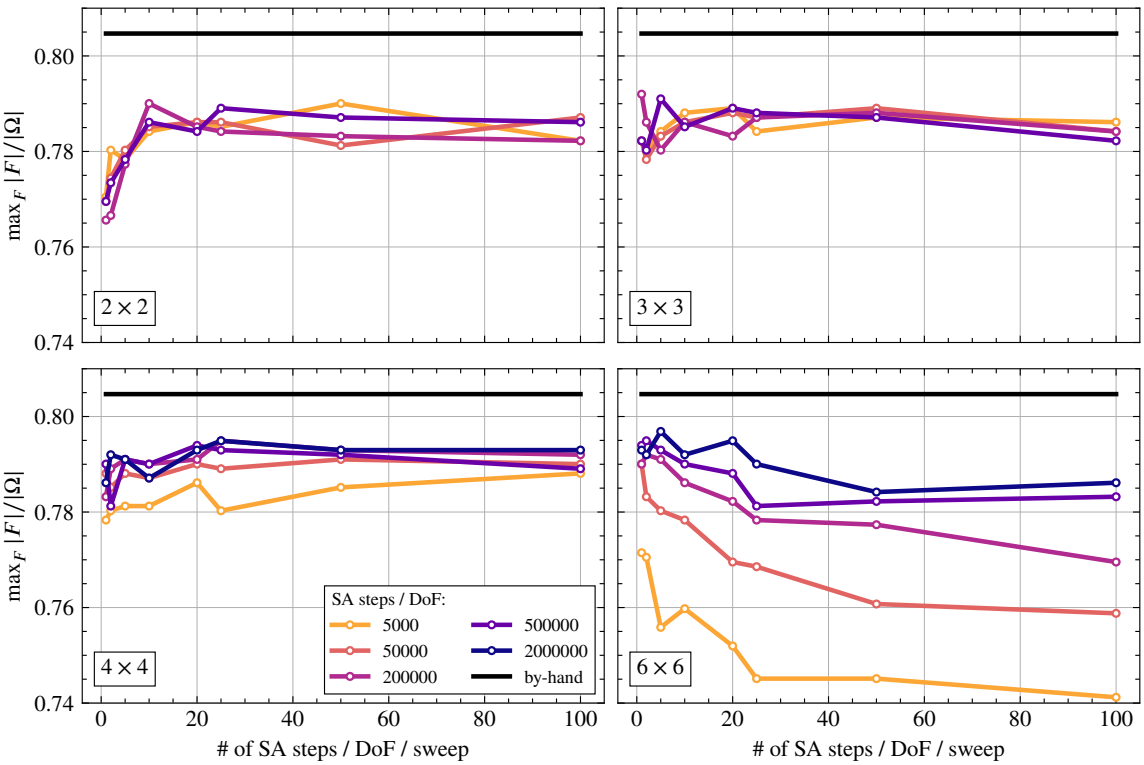


Figure 3.2: Maximum value of $|F|/|\Omega|$ with number of annealing steps per DoF per GS sweep for different numbers of annealing steps per DoF for the 32×32 uniform-grid five-point finite-difference discretization. Each panel shows a different size of geometrically chosen subdomain: 2×2 (top-left), 3×3 (top-right), 4×4 (bottom-left), 6×6 (bottom-right).

substantial work in the SA process, as shown in the top row for 2×2 and 3×3 subdomains. Here, while there is clearly a small benefit to increasing the number of SA steps per DoF per sweep from 1 to about 10, there is little improvement beyond those results, and little correlation between the quality of partitioning generated and the total amount of work invested. This occurs consistently in the numerical results throughout this paper: for small subdomain sizes, each subdomain seems to have too little freedom to make adjustments into better global configurations while still satisfying local constraints. Secondly, when we consider larger subdomains (as in the bottom row), we see that doing more work overall does, indeed, pay off, particularly for the largest subdomains (6×6, at bottom right). This is also consistently observed; in particular, that for larger subdomains we see both better overall configurations (if

sufficient work is performed) and improvements with larger work budgets. Finally, for the largest subdomains, we see a clear benefit to doing relatively few SA steps per DoF per cycle. Thus, in further results, we generally fix the number of SA steps per DoF per cycle to be relatively small, either 1 or 5.

A key question is how to balance the parameters in the SA algorithm to achieve reasonable performance at an acceptable cost. To examine this, we fix the total number of annealing steps per DoF and consider the best partitioning achieved by varying the subdomain size in the left panel of Figure 3.3. Here, we run for a number of different values of the number of SA steps per DoF per Gauss-Seidel sweep, and take the best partitioning observed over these grids for each subdomain size (noting that the optimal choice varies with subdomain size, typically being larger for small subdomain sizes and smaller for large subdomain sizes, see Table 3.10 in Appendix A for full details). When using 200 000 SA steps per DoF, there is a clear maximum in the graph for 4×4 subdomains, although the relative difference in quality in not substantial; however, when using 2 000 000 SA steps per DoF, we see continued improvement in the quality of the partitioning up to the 6×6 subdomain case. In the right panel of Figure 3.3, we look more closely at the convergence of the results with increasing numbers of SA steps per DoF for the case of 6×6 subdomains, again taking the best results obtained for different values of the number of SA steps per DoF per Gauss-Seidel sweep, see Table 3.11 in Appendix A for full details. Here, we see a clear improvement in the results up to $\mathcal{O}(10^5)$ SA steps per DoF, and continued improvement up to 2 000 000 SA steps per DoF. We recall that we fix the SA temperature reduction rate, α, so that $\alpha \to 1$ as the number of SA steps increases, yielding the same total reduction in T for each experiment. Thus, the results in Figure 3.3 are consistent with the expected behaviour of SA, that we can achieve results arbitrarily close to the global maximizer of our functional but only if we take many steps and slowly "cool" the SA iteration. For a more practical algorithm, we emphasize the behaviour at lower numbers of SA steps / DoF, noting that we achieve results within 5% of the best-known solution already with only 3000 steps / DoF, and within 2% at around 50 000 steps / DoF.

A natural question that arises from the right-hand panel of Figure 3.3 is whether the best meshes obtained occur early or late in the annealing process. That is, while we clearly see benefit from slow "cooling" of the annealing process (many SA steps per DoF), it is important to identify *when* the optimal results are obtained during the annealing process. Figure 3.4 shows the annealing history for a sample run, on the

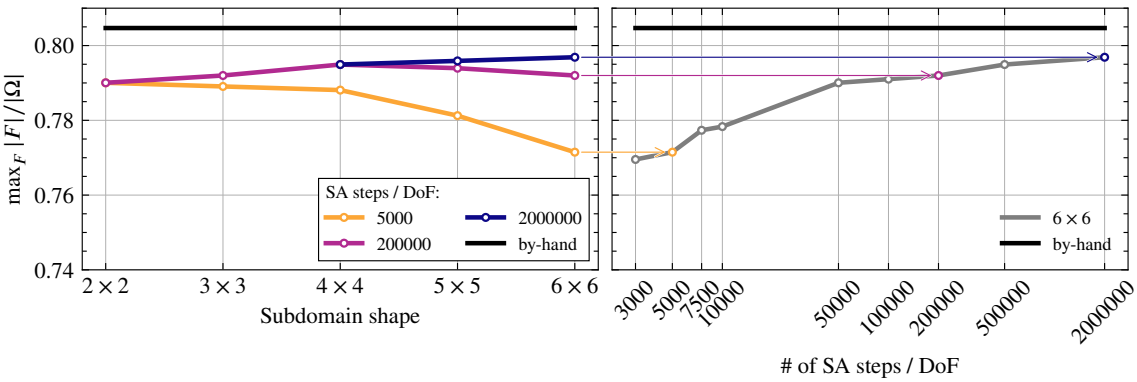


Figure 3.3: Left: Change in $|F|/|\Omega|$ with subdomain size for the 32×32 uniform-grid five-point finite-difference discretization, using geometric subdomains, with 5000, 200 000, and 2 000 000 total SA steps per DoF. Right: Change in maximum number of F-points with number of total SA steps per DoF for this problem using 6×6 subdomains.

32×32 uniform grid five-point finite-difference stencil, with 2 000 000 SA steps per DoF (for a total of almost 2×10^9 annealing steps). At right, we see the temperature decay, following an exponential curve from $T = 1.0$ at the first step to $T = 0.1$ at the final step. At left, we plot the changes in the ratio $|F|/|\Omega|$ compared to the optimized by hand mesh for this grid (also shown). For comparison, the ratio from the greedy algorithm is also shown. We see that after an initial rapid improvement in the value of the ratio, there is a secondary period where performance improves notably but steadily, up to between 5×10^8 and 10^9 total annealing steps. This demonstrates that many annealing steps are, indeed, needed to reach the best partitioning seen here, although a good partitioning is still found after many fewer steps.

Thus far, we have only considered the choice of α prescribed at the beginning of this section, with the decay rate chosen to yield a fixed temperature decay by a factor of 10 over the total number of SA steps given. In contrast, Figure 3.5 considers fixing the decay rate to be that used for 200 000 SA steps per DoF, $\alpha = 0.1^{(1.0/(200000 \times 32 \times 32))}$, but then running between 4 times fewer and 2.5 times more total SA steps, with one SA step per DoF per cycle, yielding final temperature values between 0.5623 and 0.0032. At left of Figure 3.5, we observe little correlation between the total number of SA steps per DoF and the resulting value of $|F|/|\Omega|$, with all values between 0.79 and 0.80, comparable to those seen for similar SA budgets in Figure 3.3. While this appears to offer an opportunity for saving some cost in the SA algorithm (by using

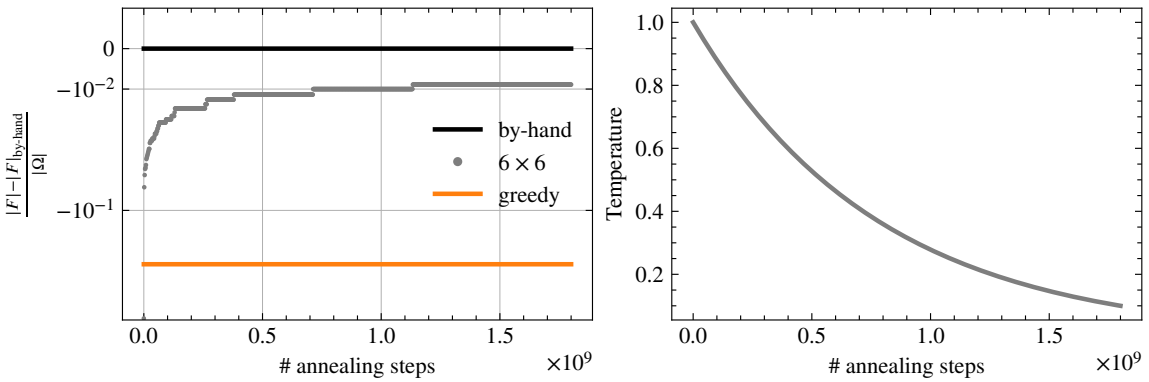


Figure 3.4: Change in $|F|/|\Omega|$ (left) and temperature (right) with number of annealing steps for the 32×32 uniform-grid finite-difference discretization of the Laplacian. At left, the case of SA with five SA steps per DoF per GS sweep using 6×6 subdomain size is shown, along with greedy as a baseline. Note that the vertical axis on the left plot uses a mixed log-linear scale for clarity, with a break at -10^{-2}.

fewer Gauss-Seidel sweeps to get comparable results), we note that using 50 000 SA steps per DoF is already prohibitively expensive for an "online cost" for this algorithm, requiring more than 30 minutes to compute the partitioning for this relatively small problem.

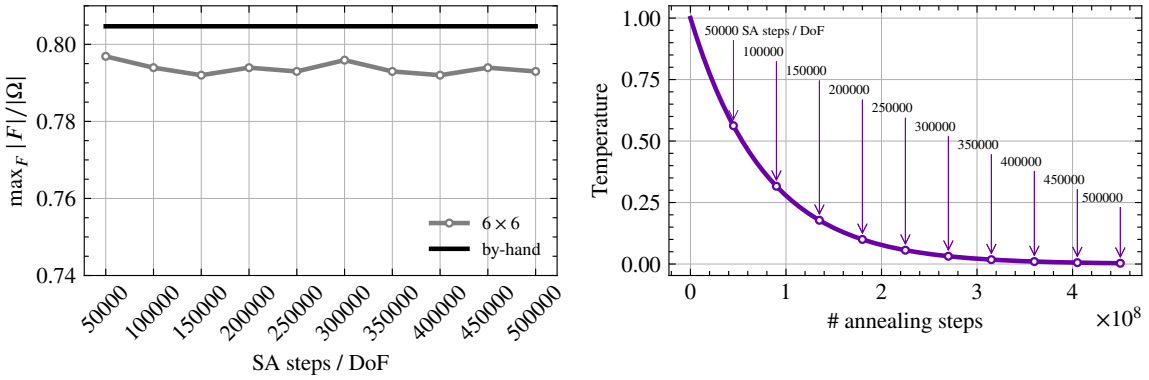


Figure 3.5: Change in $|F|/|\Omega|$ with SA steps per DoF for a fixed temperature decay rate (left) and temperature (right) for the 32×32 uniform-grid finite-difference discretization of the Laplacian. Here, one SA step per DoF per sweep is used, and the temperature decay rate is fixed, with $\alpha = 0.1^{(1.0/(200000 \times 32 \times 32))}$. The circles in the right figure show the stopping temperatures for the annotated SA steps per DoF.

We next address the nature of the grids generated by annealing, and whether they resemble grids that could be selected geometrically for this problem. Figure 3.6 shows two of the grids generated, along with the subdomains used in their generation. These represent the "best" grids found by the annealing procedure, with ratios of $|F|/|\Omega|$ of around 0.795. While these grids yield competitive ratios, there is no clear global geometric pattern, nor obvious relationship to the best optimized by hand grid shown in Figure 3.1. Furthermore, there are no clear improvements of these grids that could be readily made, such as single coarse points that could be omitted without leading to constraint violations. This suggests that the energy landscape for this problem is likely dominated by locally optimal configurations that are separated by states with constraint violations and/or sharp changes in energy.

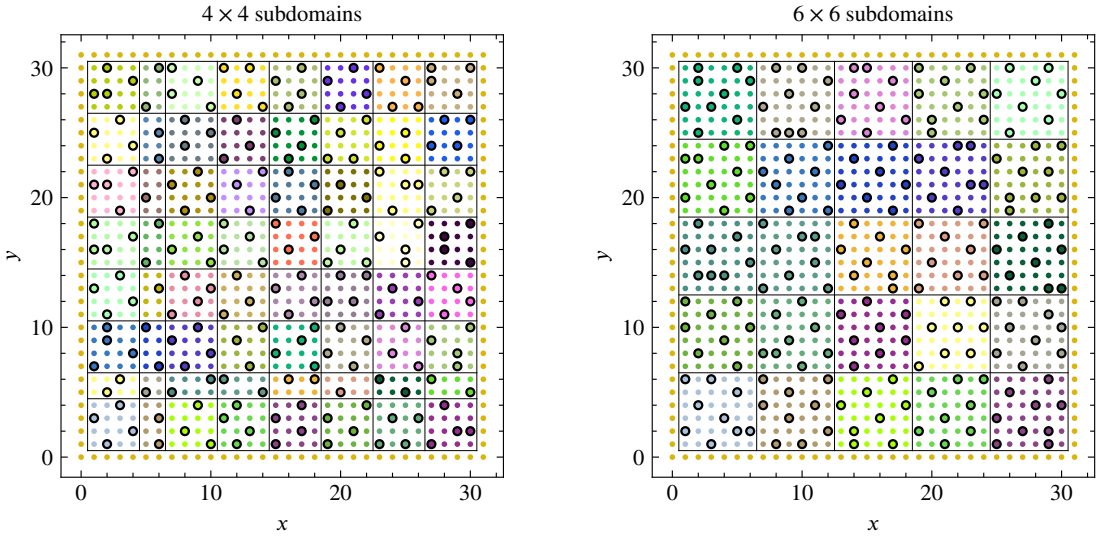


Figure 3.6: Grid partitioning for 32×32 uniform grid with five-point FD stencil using two million SA steps per DoF. At left, the partitioning is generated using primarily 4×4 subdomains, with 25 SA steps per DoF per cycle. At right, the partitioning is generated using 6×6 subdomains, with 5 SA steps per DoF per cycle. The grid at left has 814 F-points, while that at right has 816.

Figure 3.7 shows how the performance of the annealing algorithm scales with problem size, for both the geometric choice of subdomains for the finite-difference discretization discussed so far, and for the finite-element discretization. In addition, an algebraic selection (discussed below) is added for comparison. All methods (includ-

ing the optimization by hand) perform relatively well for small meshes, but degrade as the mesh size increases. The amount of work needed to achieve these results with the annealing algorithm also increases with grid size. For the 8×8 mesh, using a single (global) subdomain, equally-good partitions to the optimization by hand can be found with just 2000 annealing steps per DoF (and fewer when more subdomains are used). For the 16×16 mesh, more work and larger subdomains are needed to achieve such performance. As noted above, with small subdomains even using 200 000 annealing steps per DoF does not achieve performance equal to the by-hand partitioning on the 16×16 mesh. For larger subdomains, the algorithm does equal the results of the by-hand partitioning, but with increasing work as subdomain size increases: for 4×4 subdomains, 10 000 annealing steps per DoF are needed, while 50 000 annealing steps per DoF are needed for 5×5 subdomains. For 6×6 subdomains, even using 200 000 annealing steps per DoF, we could not recover results matching the by-hand partitioning, although we speculate that this would have occurred with even more work invested. For larger domain sizes, the results shown in Figure 3.7 represent the best results found for a given grid over all runs with varying subdomain sizes, total SA steps per DoF, and SA steps per DoF per GS sweep. While these best results are, in general, achieved with the largest allocations of SA steps per DoF tried in our experiments, the marginal benefit of considering such large amounts of work to generate the coarsenings are quite low. Here, in all cases where we invested the computational effort to explore the question, we found less than a 1% improvement in $|F|/|\Omega|$ when increasing beyond $\mathcal{O}(10^5)$ SA steps per DoF (and, in many cases, the improvement was only by one or two F-points).

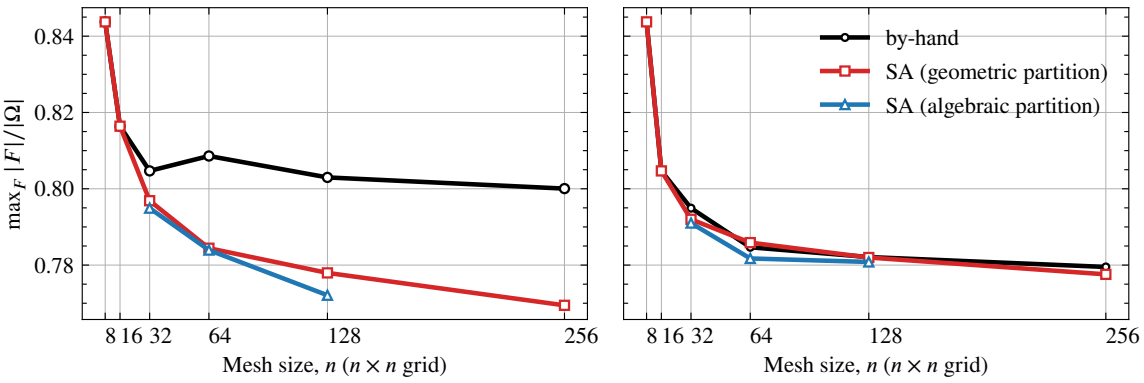


Figure 3.7: Change in $|F|/|\Omega|$ with mesh size for FD (left) and FE (right) discretizations.

For the nine-point finite-element stencil, we see similar results to those reported above and, consequently, do not include figures detailing the individual experiments in as much detail. Most notably, the best partitioning that we achieve by hand is a slightly worse in this case (as detailed in Section 3.3.4), and the partitioning using the greedy algorithm is better than in the FD case, yielding $|F|/|\Omega| = 0.752$. At left of Figure 3.8, we show an analogous figure to Figure 3.2 for the case of 5×5 subdomains. Here, we observe the same stratification. However, we are able to recover the same quality of partitioning as the best by-hand partitioning using 5×5 subdomains and 2 000 000 SA steps per DoF. Also note that we see the same mild dependence on the number of SA steps per DoF per sweep as we did in the FD case. The right panel of Figure 3.8 shows how the quality of partitioning changes with subdomain size and total work budget, with SA steps per DoF per sweep reported in Table 3.14 in Appendix A. Similarly to the FD case, there is an improvement in performance with additional work for larger subdomain sizes, but that improvement stagnates with additional work for smaller subdomain sizes. Sample grids generated for the finite-element case, including colouring to indicate subdomain choice, are shown in Figure 3.9.

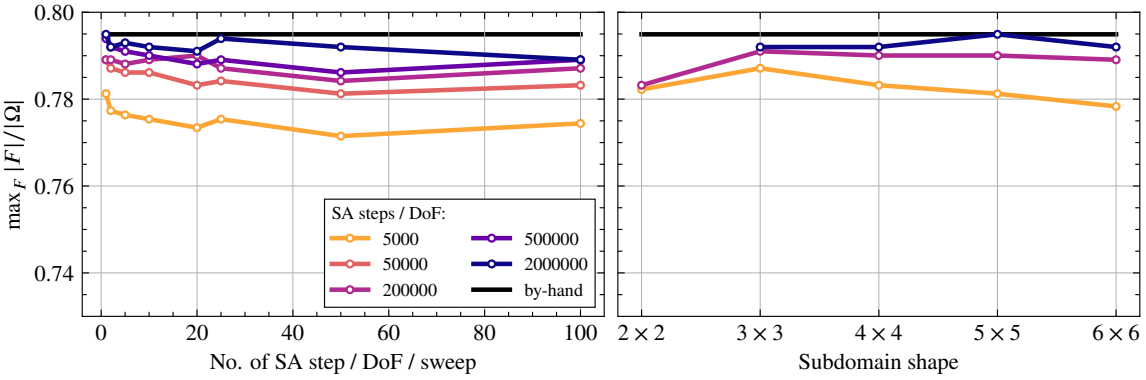


Figure 3.8: Quality of coarsening for the 32×32 uniform-grid nine-point finite-element discretization, using geometric subdomains. Left: Maximum value of $|F|/|\Omega|$ with number of annealing steps per DoF per GS sweep for different numbers of annealing steps per DoF using 5×5 geometric subdomains. Right: Change in $|F|/|\Omega|$ with subdomain size and total number of SA steps per DoF.

Finally, we verify that the result two-level AMGr algorithms are at least as effective as predicted in theory, by measuring asymptotic convergence factors (ρ), grid

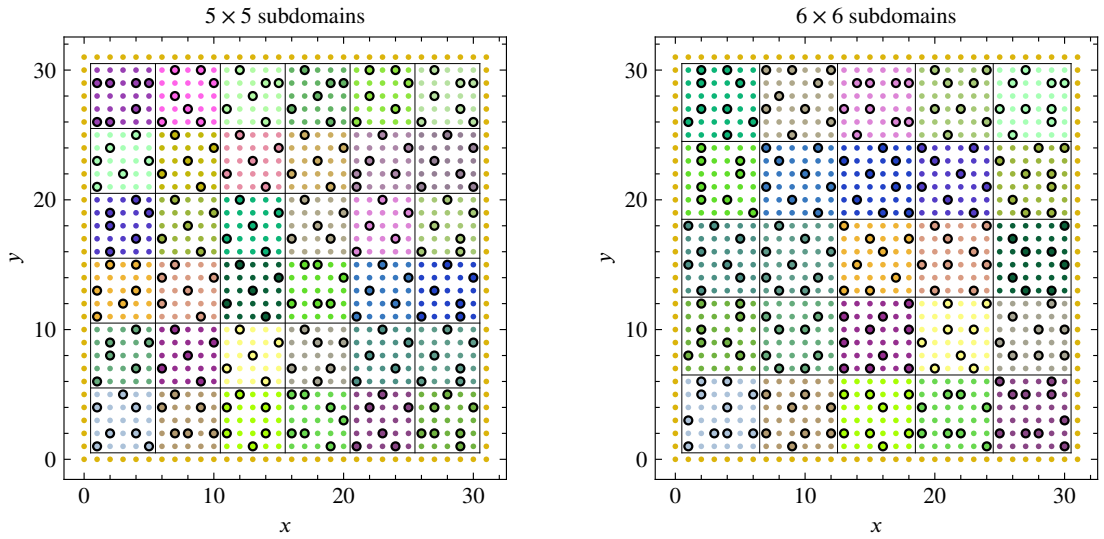


Figure 3.9: Grid partitioning for 32×32 uniform grid with nine-point FE stencil using two million SA steps per DoF. At left, the partitioning is generated using 5×5 subdomains, with one SA step per DoF per cycle. At right, the partitioning is generated using 6×6 subdomains, also with one SA step per DoF per cycle. The grid at left has 814 F-points, while that at right has 811.

complexities (C_{grid}, equal to the ratio of sum of the number of DoFs on each level of the hierarchy to that on the finest level), and operator complexities (C_{op}, equal to the ratio of the sum of the number of nonzero entries in the system matrix on each level of the hierarchy to that on the finest level). We approximate ρ by running the cycle with a random initial guess, $x^{(0)}$, and zero right-hand side, and then compute $\left(\left\|x^{(k)}\right\| / \left\|x^{(0)}\right\|\right)^{1/k}$, where $x^{(k)}$ is the approximation (to the true solution, which is the zero vector) after k cycles. Because the convergence factors of AMGr are somewhat larger than those for classical AMG, we take $k = 800$ to ensure that we sample the asymptotic behaviour suitably. Table 3.1 reports this data for two-level cycles for both the FD and FE discretizations, for both the geometric subdomain choice considered here and the algebraic subdomain choice discussed next. Considering the geometric subdomain choice, we see grid-independent convergence factors of about 0.9 for the FD case and 0.7 for the FE case. While these are notably worse than are observed for typical multigrid methods for these problems, they are consistent with the existing results for AMGr and, in particular, conform with the convergence rate bound from Theorem 3.1 of 0.977 for $\theta = 0.56$ and $\nu = 1$. Notably, neither the convergence factor nor the measured complexities degrade substantially with problem

size. We note that, in subsequent work, we have improved convergence of AMGr for these and other model problems [41].

Table 3.1: Performance of two-level AMGr on test matrices from discretizations of the 2D Laplacian.

		Geometric Subdomains			Algebraic Subdomains		
Scheme	Grid size	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}
FD	8×8	0.85	1.16	1.12	0.85	1.16	1.12
	16×16	0.86	1.19	1.18	0.86	1.19	1.18
	32×32	0.88	1.20	1.20	0.88	1.21	1.20
	64×64	0.89	1.22	1.22	0.88	1.22	1.22
	128×128	0.89	1.22	1.23	0.88	1.22	1.23
FE	8×8	0.70	1.16	1.15	0.70	1.16	1.15
	16×16	0.63	1.20	1.22	0.63	1.20	1.22
	32×32	0.67	1.21	1.23	0.69	1.21	1.25
	64×64	0.71	1.21	1.25	0.72	1.22	1.27
	128×128	0.71	1.22	1.27	0.71	1.22	1.27

3.4.2 Structured-grid discretizations with algebraically chosen subdomains

We next consider partitioning the structured-grid problems using an algebraic choice of the subdomains based on Lloyd aggregation. Figure 3.10 shows the change in the maximum number of F-points (scaled by the total number of DoFs) with the change of the subdomain size for both the FD and FE discretizations (left and right, respectively). Similarly to the case of geometric partitioning, we see relatively poor performance for small subdomain sizes, regardless of the work allocated to the SA process. For larger subdomain sizes, we see improving results with number of SA steps per DoF, as seen above. Also as seen above, the optimal subdomain size varies with total work allocation, increasing as we increase the amount of work per DoF, but even with 2 000 000 SA steps per DoF, we do not see the best performance with largest subdomains for the FE discretization. Considering variation in problem size,

Figure 3.7 shows that, as in the geometric subdomain case, we see some decrease in performance as problem size grows, but that this decrease seems to (mostly) plateau at larger problem sizes. In comparison with the geometric subdomain choice, we see some small degradation in performance with algebraically chosen subdomains, particularly in the FD case, but it is small in comparison with the optimality gap between the optimization by hand solutions and those generated by SA.

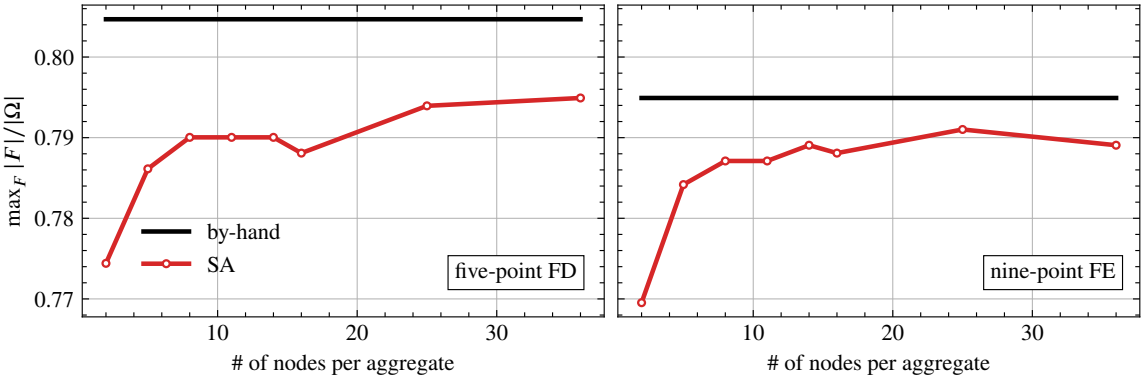


Figure 3.10: Change in maximum number of F-points with subdomain size for algebraic subdomain selection on 32×32 meshes. Left and right figures are for FD and FE schemes, respectively, showing the largest value of $|F|/|\Omega|$ attained over experiments with fixed subdomain size and varying the total number of SA steps per DoF and SA steps per sweep, as in the geometric subdomain case.

Performance of the resulting two-grid cycles is tabulated in Table 3.1, where we see very comparable convergence factors, as well as grid and operator complexities, as in the geometric subdomain case. Table 3.2 tabulates convergence factors for three-level V- and W-cycles (denoted ρ_V and ρ_W, respectively), along with three-level grid and operator complexities. For the FD case, we see some degradation in convergence when using V-cycles, while W-cycles give convergence factors similar to those in the two-level case. For the FE problem, there is less degradation for V-cycles, but still W-cycles are required to get convergence factors comparable to the two-level case. As we have increased the depth of the multigrid hierarchy, the grid and operator complexities in Table 3.2 are naturally larger than those in Table 3.1, but the growth in these complexity measures is consistent with optimally scaling multigrid methods. Performance of multilevel V- and W-cycles is also shown in Table 3.2, where coarsening is continued until the number of nodes falls below 100, yielding a four-level

cycle for the 32×32 grid, five-level cycle for 64×64, and six-level cycle for 128×128. The resulting convergence factors for both FD and FE cases remain similar to those for three-level cycles, while complexity measures grow consistently with the number of levels.

Table 3.2: Performance of three-level and multilevel AMGr on test matrices from discretizations of the 2D Laplacian.

		Three-level cycles				Multilevel cycles			
Scheme	Grid size	ρ_V	ρ_W	C_{grid}	C_{op}	ρ_V	ρ_W	C_{grid}	C_{op}
FD	32×32	0.92	0.90	1.25	1.26	0.93	0.90	1.26	1.27
	64×64	0.93	0.91	1.27	1.28	0.94	0.91	1.28	1.31
	128×128	0.94	0.91	1.28	1.30	0.95	0.92	1.30	1.35
FE	32×32	0.76	0.72	1.25	1.31	0.76	0.72	1.26	1.31
	64×64	0.76	0.73	1.27	1.35	0.77	0.73	1.28	1.37
	128×128	0.79	0.75	1.28	1.37	0.79	0.75	1.29	1.41

For the multilevel results in Table 3.2, we use 200 000 SA steps per DoF in all cases, except on the finest meshes for the 32×32 and 64×64 grids, where 2 000 000 SA steps per DoF were used. (Similar results are also seen without these "extra" steps on these problems, however.) For the 64×64 mesh, using 200 000 SA steps per DoF on all levels takes about 18 hours of "offline" time to compute the coarse meshes, in comparison to "online" solution times of only 0.003 seconds for a V-cycle and 0.011 seconds for a W-cycle. Table 3.3 presents results for the same experiment, but using only 200 and 2000 SA steps per DoF, to investigate how performance is changed when using "bad" solutions to the optimization problem in (3.3). Here, we see slight improvements in convergence factors in comparison to those in Table 3.2; however, using fewer annealing steps leads to much larger grid and operator complexities than observed above. We note here that using fewer annealing steps also leads to results that are more heavily influenced by the random nature of the SA process; here, we report complexities from runs used to generate W-cycle convergence factors, which vary slightly from those of an independent run to generate V-cycle convergence factors, but by no more than 0.01 in C_{grid} and 0.05 in C_{op}.

For our final isotropic, structured-grid test problem, we consider the FE discretiza-

Table 3.3: Performance of multilevel AMGr on test matrices from discretizations of the 2D Laplacian using 200 and 2000 SA steps per DoF.

		200 SA steps per DoF				2000 SA steps per DoF			
Scheme	Grid size	ρ_V	ρ_W	C_{grid}	C_{op}	ρ_V	ρ_W	C_{grid}	C_{op}
FD	32×32	0.89	0.86	1.42	1.62	0.91	0.88	1.33	1.44
	64×64	0.89	0.88	1.46	1.86	0.92	0.88	1.37	1.51
	128×128	0.91	0.88	1.48	2.08	0.93	0.89	1.38	1.58
FE	32×32	0.72	0.68	1.34	1.52	0.73	0.71	1.29	1.40
	64×64	0.74	0.70	1.37	1.65	0.76	0.70	1.32	1.49
	128×128	0.77	0.70	1.38	1.76	0.76	0.72	1.33	1.56

tion of the two-dimensional isotropic diffusion problem, $-\nabla \cdot \boldsymbol{K}(x,y)\nabla u(x,y) = f(x,y)$, in the domain $[0,1] \times [0,1]$ with Dirichlet boundary conditions and piecewise constant ("jumping") coefficient, $\boldsymbol{K}(x,y)$. To determine $\boldsymbol{K}(x,y)$, we select 20% of the elements at random, and set $\boldsymbol{K}(x,y) = 10^{-8}$ in these elements, with value 1 in the remaining 80% of the elements, matching one of the test problems from [29]. We partition the nodes using 200 000 SA steps per DoF, and show the first coarse mesh chosen using 1 SA step per DoF per sweep for the 31×31 grid in Figure 3.11. Convergence factors and grid and operator complexities for multilevel V- and W-cycles are shown in Table 3.4. We note, in particular, that these results are quite comparable to those shown in Table 3.2 for the case of $\boldsymbol{K}(x,y) = 1$ everywhere. In comparison to the results presented in [29], we see larger convergence factors here (0.78 for the 127×127 grid, in comparison to 0.59 reported there), but with lower complexities ($C_{\text{op}} = 1.42$ here, compared to 1.75 there).

Table 3.4: Performance of multilevel AMGr on test matrices from discretizations of the 2D Laplacian with jumping coefficients.

Scheme	Grid size	ρ_V	ρ_W	C_{grid}	C_{op}
FE	31×31	0.73	0.71	1.27	1.33
	63×63	0.76	0.73	1.29	1.38
	127×127	0.78	0.76	1.29	1.42

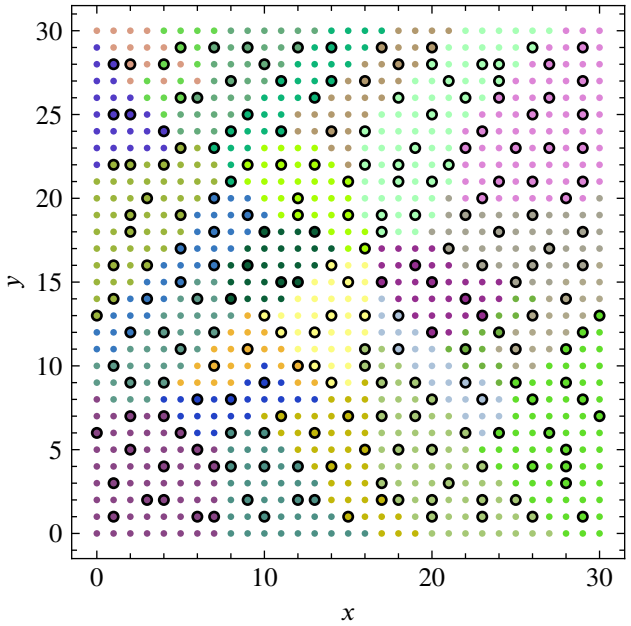


Figure 3.11: Partitioning for the isotropic problem on a 31×31 uniform grid with jumping coefficients, generated using 200 000 SA steps per DoF, 1 SA step per DoF per GS sweep, and subdomains with an average of 36 points per subdomain. C-points are represented by the black circles.

3.4.3 Anisotropic problems on structured grids

Next, we consider the two-dimensional anisotropic diffusion problem, $-\nabla \cdot \boldsymbol{K}(x,y)\nabla u(x,y) = f(x,y)$, in the domain $[0,1] \times [0,1]$ with Dirichlet boundary conditions. We choose the tensor coefficient $\boldsymbol{K}(x,y) = QMQ^T$, where $Q = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$, and $M = \begin{bmatrix} \delta & 0 \\ 0 & 1 \end{bmatrix}$. The parameters $0 < \delta \leq 1$ and θ specify the strength and direction of anisotropy in the problem, respectively, with $\theta = 0$ giving the anisotropic problem $-\delta u_{xx} - u_{yy} = f$ and $\theta = \pi/2$ giving $-u_{xx} - \delta u_{yy} = f$. For $0 < \theta < \pi/2$, the axis of the small diffusion coefficient in the problem rotates clockwise from being in the positive x-direction for small θ to the positive y-direction for $\theta \approx \pi/2$. Anisotropic problems cause difficulty for the greedy coarsening algorithm [29], where large grid and operator complexities and poor algorithmic performance were overcome by augmenting the coarse grids with the second pass of the Ruge-Stüben coarsening algorithm and using classical AMG interpolation in place of the AMGr interpolation operator. We

emphasize that both the FD and FE discretizations of this problem result in non-diagonally dominant system matrices and, consequently, the convergence guarantee from Theorem 3.1 does not apply in this setting. We include these problems, nonetheless, both to stress-test our approach and highlight the need for further research in this direction.

As an example of a problem in this class, we fix $\delta = 10^{-6}$ and $\theta = \pi/3$. Figure 3.12 shows the coarse-grid points selected for the bilinear finite-element discretization of the problem on a uniform 32×32 mesh. At left, we see that the partitioning generated by the SA algorithm correctly detects that this is an anisotropic problem with strong coupling primarily in the x-direction and weak coupling primarily in the y-direction, producing grids that are consistent with semi-coarsening, with some regions of coarsening along diagonals. Unfortunately, however, this grid leads to poor convergence using AMGr interpolation. As in the original experiments by MacLachlan and Saad [29], each fine-grid point has multiple coarse-grid neighbours (leading to an increase in operator complexity, due to the nonzero structure of $D_{FF}^{-1}A_{FC}$), but since A is no longer diagonally dominant in the anisotropic case, the assumption that $\begin{bmatrix} D_{FF} & -A_{FC} \\ -A_{FC}^T & A_{CC} \end{bmatrix}$ is positive semi-definite is violated, and the resulting performance is poor. To overcome this failure, we augment the coarse-grid set using the second-pass algorithm from Ruge-Stüben AMG, using the classical strength of connection parameter of 0.30 to determine strong connections in the graph. At right of Figure 3.12, we show the resulting graph, which now satisfies the requirement that every pair of strongly connected F-points has a common C-neighbour (which is clearly not satisfied in the partitioning at left). Unfortunately, this comes at a heavy cost, as the grid at right now has many more C-points than we would like. Below, we verify that these grids lead to effective multigrid hierarchies, when coupled with classical AMG interpolation, but note the increased grid and operator complexities in these hierarchies. A key question for future work (addressed in [41]) is whether we can determine algebraic interpolation operators for grids such as those at left that lead to effective AMGr performance.

Tables 3.5 and 3.6 present convergence factors and grid and operator complexities for two- and three-level cycles, respectively. For geometric partitioning, we use 200 000 SA steps per DoF with one or two SA steps per DoF per cycle and 5×5 or 6×6 subdomains (depending on what worked best for a given problem, reported in Table 3.16 in Appendix A). Similar choices are made using algebraic partitioning,

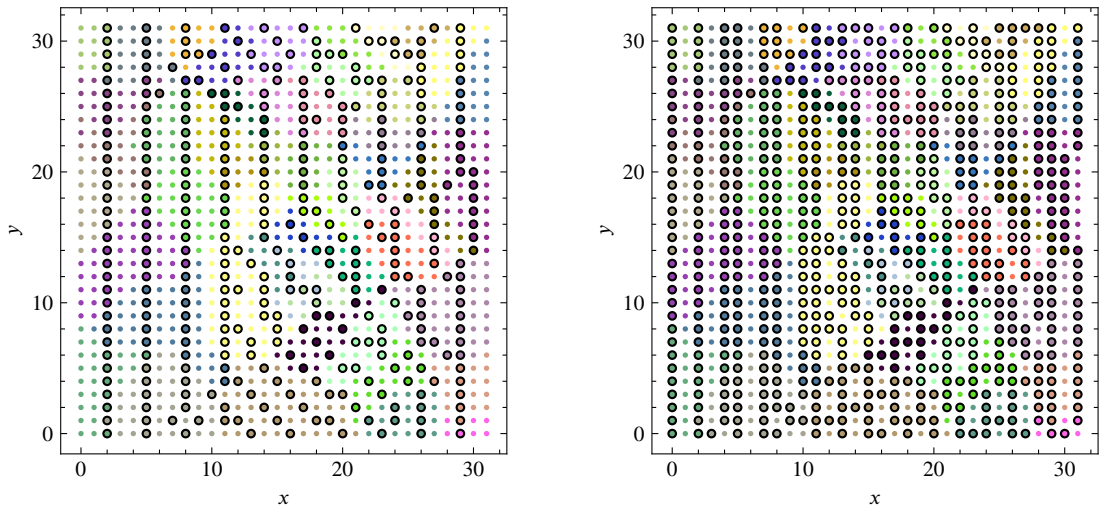


Figure 3.12: Grid partitioning for an anisotropic diffusion problem with $\delta = 10^{-6}$ and $\theta = \pi/3$, discretized on a 32×32 mesh using the bilinear FE stencil. The initial partitioning, at left, is generated using algebraic subdomains averaging 20 points per subdomain, using 2 000 000 SA steps per DoF and 1 SA step per DoF per GS sweep, and has 664 F-points and 360 C-points. At right is the grid augmented using the second pass of the classical AMG algorithm, resulting in 343 F-points and 681 C-points.

although we see slight improvements using 2 000 000 SA steps per DoF for some problems. For the three-level tests, we uniformly use 200 000 SA steps per DoF, with 1 SA step per DoF per GS sweep, and algebraic subdomains with average size of 36 points. In Table 3.5, we again see that there is relatively little difference in results generated using geometric and algebraic choices of the Gauss-Seidel subdomains. Notably, both choices lead to effective cycles for both the finite-difference and finite-element discretizations, albeit at the cost of increased grid and operator complexities. Table 3.6 again shows degradation in performance moving from two-grid to three-grid V-cycles, but that three-grid W-cycles mostly recover comparable convergence to the two-level case, with some notable degradation for the finite-element operator.

3.4.4 Discretizations on unstructured grids

We next consider two diffusion problems with homogeneous Dirichlet boundary conditions on unstructured triangulations of square domains, discretized using linear finite elements. First, we consider an isotropic diffusion operator ($-\nabla \cdot \nabla u = f$) on $[-1, 1]^2$,

Table 3.5: Performance of two-level AMGr for the anisotropic diffusion problem with $\delta = 10^{-6}$ and $\theta = \pi/3$ on structured meshes.

		Geometric Partitioning			Algebraic Partitioning		
Scheme	Grid size	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}
FD	32×32	0.58	1.70	2.09	0.57	1.70	2.09
	64×64	0.58	1.71	2.11	0.58	1.71	2.11
	128×128	0.60	1.72	2.12	0.63	1.72	2.12
FE	32×32	0.62	1.65	2.02	0.61	1.67	2.00
	64×64	0.61	1.67	2.09	0.62	1.67	2.08
	128×128	0.61	1.67	2.11	0.61	1.67	2.12

Table 3.6: Performance of three-level AMGr for the anisotropic diffusion problem with $\delta = 10^{-6}$ and $\theta = \pi/3$ on structured meshes.

Scheme	Grid size	ρ_V	ρ_W	C_{grid}	C_{op}
FD	32×32	0.72	0.57	2.16	3.13
	64×64	0.76	0.60	2.16	3.20
	128×128	0.80	0.66	2.18	3.26
FE	32×32	0.74	0.61	2.06	2.86
	64×64	0.81	0.69	2.07	3.05
	128×128	0.89	0.82	2.05	3.10

generated by taking an initially unstructured mesh, refining it a set number of times and, then, smoothing the resulting mesh. We consider three levels of refinement for this example, resulting in meshes with 1433, 5617, and 22 241 DoFs. The SA-based partitioning scheme appears to perform similarly well in this setting, with a splitting found for the smallest mesh shown in Figure 3.13. While we no longer have hand-generated estimates of the optimal coarsening factors, we note that the SA-based partitioning scheme outperforms the greedy algorithm, generating F-sets with 1106, 4241, and 16 604 points, for these three grids, respectively, in comparison to F-sets of 1024, 3916, and 15 079 points. As above, this improvement comes at a cost: the offline time to partition the 5617 DoF problem using 200 000 SA steps per DoF is over 12 hours for the two-level scheme. Two- and three-level convergence factors, as

well as grid and operator complexities for these systems are given in Table 3.7. Here, we observe that these results are quite similar to those seen for the structured-grid discretizations above, with some degradation in convergence seen for the three-level V-cycle results, but not in the W-cycle results. Here, the three-level results are computed using 200 000 SA steps per DoF, with 1 SA step per DoF per GS sweep, on subdomains with an average of 36 points per subdomain. Slight modifications to these parameters yield small improvements in two-level results; as a result, we use these choices in the table and report them in Table 3.17 in Appendix A.

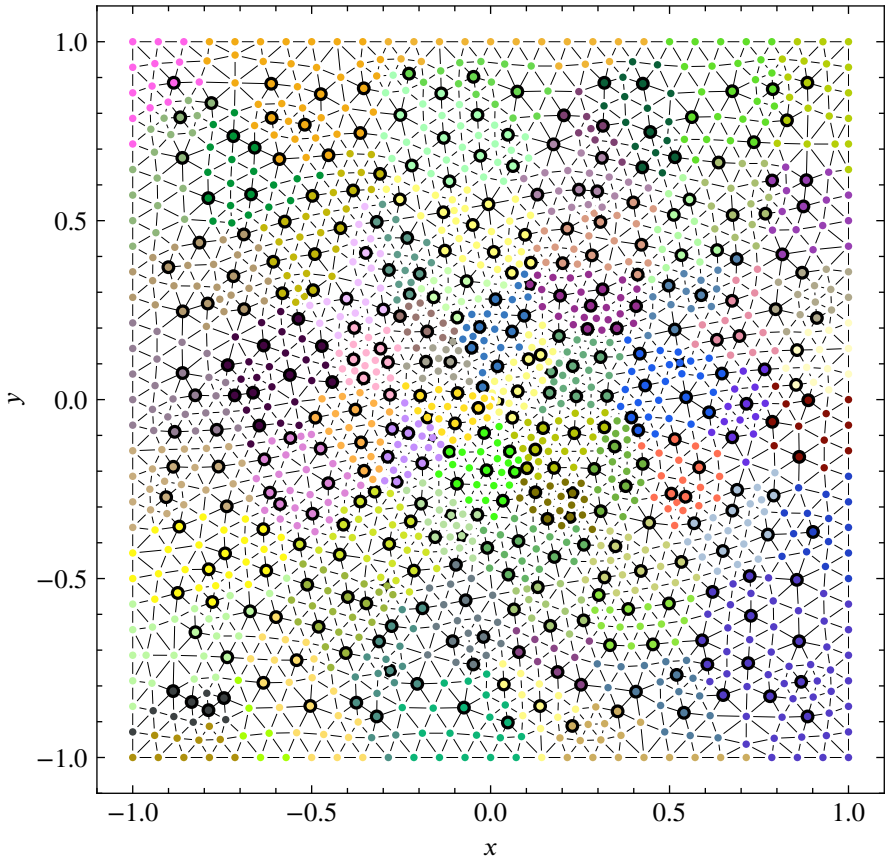


Figure 3.13: Partitioning for the isotropic problem on the unstructured mesh with 1433 points, generated using 1 000 000 SA steps per DoF, 5 SA steps per DoF per GS sweep, and subdomains with an average of 20 points per subdomain.

We next consider the anisotropic diffusion operator (again with Dirichlet boundary conditions) considered above, with $\delta = 0.01$ and $\theta = \pi/3$, on an unstructured

Table 3.7: Performance of two- and three-level AMGr for isotropic problem on unstructured meshes.

	Two-level cycle			Three-level cycles			
#DoF	ρ	C_{grid}	C_{op}	ρ_V	ρ_W	C_{grid}	C_{op}
1433	0.66	1.23	1.31	0.78	0.65	1.28	1.39
5617	0.71	1.25	1.32	0.79	0.70	1.30	1.42
22241	0.75	1.25	1.33	0.84	0.75	1.32	1.45

triangulation of the unit square taken from Brannick and Falgout [9], matching the problem labelled 2D-M2-RLap in that paper. As in Brannick and Falgout [9], we consider three levels of refinement of the mesh, with 798, 3109, and 12 273 DoFs, respectively. Figure 3.14 shows two different partitions for this mesh. At left, we give the partitioning generated using the SA-based partitioning algorithm proposed here, and at right we give the splitting after a second pass of the Ruge-Stüben coarsening algorithm where strength of connection is computed using the classical strength parameter of 0.55. As with the structured-grid case above, we found the second pass is needed to achieve good convergence factors for the anisotropic problem, but that it does so at the expense of coarsening at a much slower rate. Convergence factors, grid complexities, and operator complexities for these problems are reported in Table 3.8. As above, the complexities are higher for the anisotropic operators than the isotropic (due to the use of the second pass). Here, we observe degradation in convergence with grid refinement, although again with less degradation for W-cycles than V-cycles. The three-level results are computed with the same parameters as for the isotropic problem above, with more SA steps per DoF used for the two-level results, as this yielded slight improvements. While the performance reported in Table 3.8 is far from optimal, we note that the convergence factors for the larger two meshes are better than those reported for compatible relaxation [9], while the operator complexities are comparable to those reported therein for BoomerAMG [21]. We again emphasize that these anisotropic diffusion problems do not satisfy the convergence bounds stated above and, consequently, are included to stress-test the framework presented here; improved results for this problem are presented in [41].

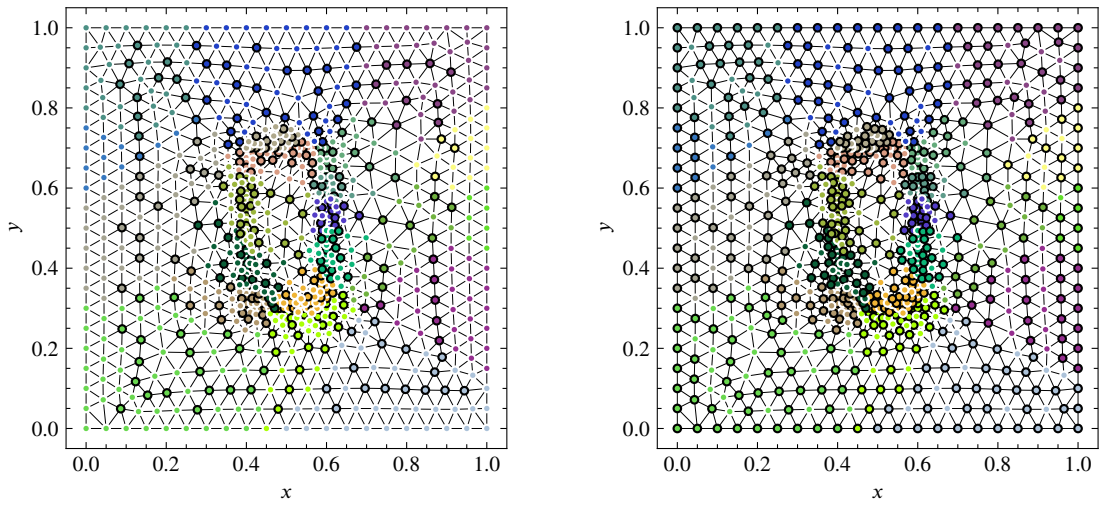


Figure 3.14: Partitioning for the anisotropic problem on an unstructured mesh containing 798 points. The partitioning at left was generated using 500 000 SA steps per DoF, with 1 SA step per DoF per GS sweep, on subdomains with an average size of 36 points per subdomain, yielding 524 F-points and 274 C-points. At right, this partitioning is augmented by the second pass of classical AMG coarsening, resulting in 263 F-points and 535 C-points.

Table 3.8: Performance of two-level and three-level AMGr for anisotropic problem on unstructured meshes.

	Two-level cycle			Three-level cycles			
#DoF	ρ	C_{grid}	C_{op}	ρ_V	ρ_W	C_{grid}	C_{op}
798	0.69	1.67	1.82	0.84	0.71	2.11	2.41
3109	0.75	1.67	1.87	0.84	0.75	2.09	2.52
12273	0.82	1.67	1.89	0.90	0.83	2.08	2.56

3.4.5 Convection-diffusion problems on structured grids

For our final tests, we consider the upwind finite-difference discretization of two singularly perturbed convection-diffusion equations. While these problems lead to nonsymmetric discretization matrices (and, as such, Theorem 3.1 no longer applies), they represent a plausible set of test problems for reduction-based methods due to their "nearly triangular" M-matrix structure [30]. In particular, we consider the solution

of the convection-diffusion equation,

$$-\varepsilon\Delta u + \boldsymbol{b}\cdot\nabla u = f,$$

with homogeneous Dirichlet boundary conditions, for two choices of the convection direction:

$$\text{grid-aligned convection } \boldsymbol{b} = \begin{bmatrix}1\\0\end{bmatrix},$$

$$\text{non-grid-aligned convection } \boldsymbol{b} = \begin{bmatrix}2\\3\end{bmatrix}.$$

We discretize these problems on uniform $N \times N$ meshes with the standard 5-point finite-difference for the diffusion term, and a first-order upwind finite-difference discretization for the convection terms. In all experiments, we use 200 000 SA steps per DoF to generate the partitioning, and generate the AMGr interpolation operator by computing $\begin{bmatrix}D_{FF}^{-1}A_{FC}\\ I\end{bmatrix}$. We construct the restriction operator by taking $\begin{bmatrix}A_{CF}D_{FF}^{-1} & I\end{bmatrix}$. Table 3.9 reports measured asymptotic convergence factors, along with grid and operator complexities for these problems, showing insensitivity to both problem size and the singular perturbation parameter, ε. Figure 3.15 shows two sample partitioning, one for each convection direction, illustrated at $\varepsilon = 10^{-5}$ for $N = 32$.

3.5 Conclusions and future work

While existing heuristics for coarse-grid selection within AMG offer reliable performance for a wide class of problems, there remains much interest in both improving our understanding of AMG convergence and developing new approaches that offer guarantees of convergence for even wider classes of problems. We believe a promising, yet under-explored, area of research is in the extension of reduction-based AMG approaches, that have been shown to be effective for both isotropic diffusion problems [28, 29] and more interesting classes of problems, such as some hyperbolic PDEs [30]. In this paper, we propose a new coarsening algorithm for AMGr, based on applying simulated annealing to the optimization problem first posed by MacLachlan and Saad [29]. We choose SA as it is a widely used optimization technique that can be

Table 3.9: Performance of two-level AMGr for convection-diffusion problems on structured meshes.

		Grid-Aligned			Non-Grid-Aligned		
ε	N	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}
	16	0.617	1.50	1.93	0.617	1.34	1.65
10^{-3}	32	0.617	1.50	2.01	0.617	1.36	1.71
	64	0.617	1.51	2.02	0.617	1.36	1.74
	16	0.617	1.50	1.93	0.617	1.34	1.63
10^{-4}	32	0.617	1.50	1.97	0.617	1.36	1.71
	64	0.617	1.51	2.03	0.617	1.36	1.74
	16	0.617	1.50	1.92	0.617	1.33	1.63
10^{-5}	32	0.617	1.51	2.00	0.617	1.36	1.71
	64	0.617	1.51	2.03	0.617	1.36	1.73

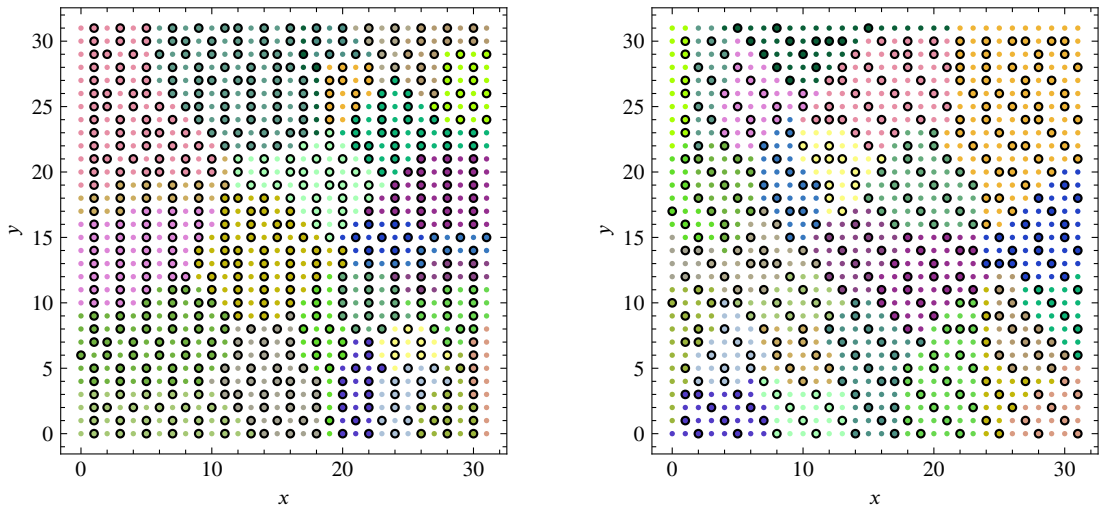


Figure 3.15: Partitioning for convection-diffusion problems on uniform grids, for the grid-aligned case, at left, and the non-grid-aligned case, at right. In both cases, we depict the partitioning for $\varepsilon = 10^{-5}$ and $N = 32$, generated using 200 000 SA steps per DoF.

applied to combinatorial optimization problems. Other techniques are certainly possible (e.g., genetic algorithms or particle swarm optimization), and it may be that those

techniques lead to improved performance. For isotropic problems on both structured and unstructured meshes, we find that the SA-based partitioning approach outperforms the original greedy algorithm, sometimes dramatically, while sharing some of the existing limitations of the AMGr framework, particularly for anisotropic problems. We believe that the performance difference between the greedy and SA-based algorithm is, simply, due to the complex nature of the optimization problem at hand. The greedy algorithm makes the best "local" choice at each stage, but those local choices force poor global configurations. In particular, the greedy approach has no opportunity to undo a choice already made, while the SA approach allows that to happen. Nonetheless, we see this as an important proof-of-concept, showing that randomized search and other derivative-free optimization algorithms can be successfully applied to the combinatorial optimization problems in AMG coarsening.

The main drawback of this approach is the high computational cost of simulated annealing. For example, computing a single coarse grid for a 32×32 mesh with 200 000 SA steps per DoF required around 2.5 hours on a modern workstation, primarily for evaluating the fitness functional in the optimization, with expected scaling in problem size and number of SA steps per DoF. Two key questions that this raises are whether the fitness functional can be approximated in a more efficient manner (e.g., using a value neural network) and whether other optimization techniques that rely on fewer samplings of the fitness functional can be applied. Both of these are topics of our current research.

This research also exposes a known weakness of the AMGr methodology (and, to our knowledge, one of all AMG-like algorithms with guarantees on convergence rates) when applied to problems that are not diagonally dominant (such as finite-element discretization of anisotropic diffusion equations). Another current research direction is identifying whether the diagonal choice of D_{FF} can be generalized to yield better convergence (while retaining a guaranteed convergence rate), and whether such changes can be accommodated within the optimization problems solved here. Preliminary results in this direction are presented in Chapter 4.

Acknowledgments

The work of S.P.M. was partially supported by an NSERC Discovery Grant. This work does not have any conflicts of interest.

Bibliography

[1] David M. Alber and Luke N. Olson. Parallel coarse-grid selection. *Numerical Linear Algebra with Applications*, 14(8):611–643, 2007. URL `http://dx.doi.org/10.1002/nla.541`.

[2] Satish Balay, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.0.0, Argonne National Laboratory, 2008.

[3] Satish Balay, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page. Technical report, 2010. http://www.mcs.anl.gov/petsc.

[4] WN Bell. *Algebraic multigrid for discrete differential forms*. Ph.D. thesis, University of Illinois at Urbana-Champaign, 2008.

[5] A. Brandt, S. F. McCormick, and J. W. Ruge. Algebraic multigrid (AMG) for automatic multigrid solutions with application to geodetic computations. Technical report, Inst. for Computational Studies, Fort Collins, Colo., 1982.

[6] A. Brandt, S. F. McCormick, and J. W. Ruge. Algebraic multigrid (AMG) for sparse matrix equations. In D. J. Evans, editor, *Sparsity and Its Applications*, pages 257–284. Cambridge University Press, Cambridge, 1984.

[7] J. Brannick, M. Brezina, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge. An energy-based AMG coarsening strategy. *Numerical Linear Algebra with Applications*, 13:133–148, 2006.

[8] J. Brannick, Y. Chen, J. Kraus, and L. Zikatanov. Algebraic multilevel preconditioners for the graph Laplacian based on matching in graphs. *SIAM Journal on Numerical Analysis*, 51(3):1805–1827, 2013. doi: 10.1137/120876083.

[9] James J. Brannick and Robert D. Falgout. Compatible relaxation and coarsening in algebraic multigrid. *SIAM Journal on Scientific Computing*, 32(3):1393–1416, 2010.

[10] Oliver Bröker. *Parallel Multigrid Methods using Sparse Approximate Inverses*. Ph.D. thesis, Swiss Federal Institute of Technology, Zurich, Zurich, Switzerland, 2003.

[11] Edmond Chow. An unstructured multigrid method based on geometric smoothness. *Numerical Linear Algebra with Applications*, 10:401–421, 2003.

[12] A. J. Cleary, R. D. Falgout, V. E. Henson, and J. E. Jones. Coarse-grid selection for parallel algebraic multigrid. In *Proc. of the Fifth International Symposium on Solving Irregularly Structured Problems in Parallel*, volume 1457 of *Lecture Notes in Computer Science*, pages 104–115. Springer–Verlag, New York, 1998.

[13] Hans De Sterck, Ulrike Meier Yang, and Jeffrey J. Heys. Reducing complexity in parallel algebraic multigrid preconditioners. *SIAM Journal on Matrix Analysis and Applications*, 27(4):1019–1039, 2006. ISSN 0895-4798.

[14] Hans De Sterck, Robert D. Falgout, Joshua W. Nolting, and Ulrike Meier Yang. Distance-two interpolation for parallel algebraic multigrid. *Numerical Linear Algebra with Applications*, 15(2-3):115–139, 2008. ISSN 1070-5325.

[15] J. de Vicente, J. Lanchares, and R. Hermida. Placement by thermodynamic simulated annealing. *Physics Letters A*, 317(5):415–423, 2003. doi: https://doi.org/10.1016/j.physleta.2003.08.070.

[16] R.D. Falgout and U.M. Yang. Hypre: A library of high performance preconditioners. In *Computational Science - ICCS 2002: International Conference, Amsterdam, The Netherlands, April 21-24, 2002. Proceedings, Part III*, volume 2331 of *Lecture Notes in Computer Science*, pages 632–641. Springer-Verlag, 2002.

[17] Astrid Franz, Karl Heinz Hoffmann, and Peter Salamon. Best possible strategy for finding ground states. *Physical Review Letters*, 86:5219–5222, 2001. doi: 10.1103/PhysRevLett.86.5219.

[18] M.W. Gee, C.M. Siefert, J.J. Hu, R.S. Tuminaro, and M.G. Sala. ML 5.0 smoothed aggregation user's guide. Technical Report SAND2006-2649, Sandia National Laboratories, 2006.

[19] Florian Gossler and Reinhard Nabben. On AMG methods with F-smoothing based on Chebyshev polynomials and their relation to AMGr. *Electronic Transactions on Numerical Analysis*, 45:146–159, 2016.

[20] V. Granville, M. Krivanek, and J.P. Rasson. Simulated annealing: A proof of convergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):652–656, 1994. doi: 10.1109/34.295910.

[21] V.E. Henson and U.M. Yang. BoomerAMG: A parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, 41:155–177, 2002.

[22] Michael A. Heroux, Roscoe A. Bartlett, Vicki E. Howle, Robert J. Hoekstra, Jonathan J. Hu, Tamara G. Kolda, Richard B. Lehoucq, Kevin R. Long, Roger P. Pawlowski, Eric T. Phipps, Andrew G. Salinger, Heidi K. Thornquist, Ray S. Tuminaro, James M. Willenbring, Alan Williams, and Kendall S. Stanley. An overview of the Trilinos project. *ACM Trans. Math. Softw.*, 31(3):397–423, 2005. ISSN 0098-3500. doi: http://doi.acm.org/10.1145/1089014.1089021.

[23] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer, 6th edition, 2018.

[24] O. Livne. Coarsening by compatible relaxation. *Numerical Linear Algebra with Applications*, 11(2):205–227, 2004.

[25] Stuart Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[26] S. MacLachlan and L. Olson. Theoretical bounds for algebraic multigrid performance: Review and analysis. *Numerical Linear Algebra with Applications*, 21 (2):194–220, 2014.

[27] S. MacLachlan and Y. Saad. Greedy coarsening strategies for nonsymmetric problems. *SIAM Journal on Scientific Computing*, 29(5):2115–2143, 2007.

[28] S. MacLachlan, T. Manteuffel, and S. McCormick. Adaptive reduction-based AMG. *Numerical Linear Algebra with Applications*, 13:599–620, 2006.

[29] Scott MacLachlan and Yousef Saad. A greedy strategy for coarse-grid selection. *SIAM Journal on Scientific Computing*, 29(5):1825–1853, 2007. doi: 10.1137/060654062.

[30] Thomas A. Manteuffel, Steffen Münzenmaier, John Ruge, and Ben Southworth. Nonsymmetric reduction-based algebraic multigrid. *SIAM Journal on Scientific Computing*, 41(5):S242–S268, 2019. doi: 10.1137/18M1193761.

[31] Artem Napov and Yvan Notay. An algebraic multigrid method with guaranteed convergence rate. *SIAM Journal on Scientific Computing*, 34(2):A1079–A1109, 2012. doi: 10.1137/100818509.

[32] L. N. Olson and J. B. Schroder. PyAMG: Algebraic multigrid solvers in Python v4.0. Technical report, 2018. URL `https://github.com/pyamg/pyamg`. Release 4.0.

[33] Luke N. Olson, Jacob B. Schroder, and Raymond S. Tuminaro. A new perspective on strength measures in algebraic multigrid. *Numerical Linear Algebra with Applications*, 17(4):713–733, August 2010. doi: 10.1002/nla.669. URL `http://dx.doi.org/10.1002/nla.669`.

[34] M. Ries, U. Trottenberg, and G. Winter. A note on MGR methods. *Linear Algebra and its Applications*, 49:1–26, 1983.

[35] J. W. Ruge and K. Stüben. Algebraic multigrid (AMG). In S. F. McCormick, editor, *Multigrid Methods*, volume 3 of *Frontiers in Applied Mathematics*, pages 73–130. SIAM, Philadelphia, PA, 1987.

[36] J. Schneider and S. Kirkpatrick. *Stochastic Optimization*. Springer, 2006.

[37] K. Stüben. An introduction to algebraic multigrid. In U. Trottenberg, C. Oosterlee, and A. Schüller, editors, *Multigrid*, pages 413–528. Academic Press, London, 2001.

[38] Paul N. Swarztrauber. The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle. *SIAM Review*, 19(3):490–501, 1977. ISSN 0036-1445. doi: 10.1137/1019071.

[39] Ali Taghibakhshi, Scott MacLachlan, Luke Olson, and Matthew West. Optimization-based algebraic multigrid coarsening using reinforcement learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 12129–12140. Curran Associates, Inc., 2021.

[40] Petr Vaněk, Jan Mandel, and Marian Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56(3): 179–196, 1996.

[41] Tareq Zaman, Nicolas Nytko, Ali Taghibakhshi, Scott MacLachlan, Luke Olson, and Matthew West. Generalizing reduction-based algebraic multigrid. 2022. Submitted; arXiv preprint 2212.08317.

Appendix A: Additional Data

In this section, we provide some additional tables to provide more complete data on the experiments reported above.

SA steps / DoF	2×2	3×3	4×4	5×5	6×6
5000	50	20	100	2	1
200 000	10	1	25	5	2
2 000 000	–	–	25	20	5

Table 3.10: Number of SA steps per DoF per sweep used for numerical results in left-hand panel of Figure 3.3.

SA steps / DoF:	3000	5000	7500	10 000	50 000	100 000	200 000	500 000	2 000 000
SA steps / DoF / cycle:	1	1	1	2	1	2	2	2	5

Table 3.11: Number of SA steps per DoF per sweep used for numerical results in right-hand panel of Figure 3.3.

Partitioning		FD Discretization					
		8×8	16×16	32×32	64×64	128×128	256×256
Geometric	SA steps per DoF	1000	10 000	2 000 000	2 000 000	200 000	200 000
	SA steps per DoF per cycle	1	100	5	1	1	1
	Subdomain size	2×2	4×4	6×6	6×6	6×6	6×6
Algebraic	SA steps per DoF	–	–	2 000 000	2 000 000	200 000	–
	SA steps per DoF per cycle	–	–	2	1	1	–
	Subdomain size	–	–	36	36	36	–

Table 3.12: Number of SA steps and SA steps per DoF per sweep used for numerical results in left-hand panel of Figure 3.7.

Partitioning		FE Discretization					
		8×8	16×16	32×32	64×64	128×128	256×256
Geometric	SA steps per DoF	1000	200 000	2 000 000	1 000 000	200 000	200 000
	SA steps per DoF per cycle	25	25	50	100	1	1
	Subdomain size	2×2	3×3	4×4	3×3	5×5	5×5
Algebraic	SA steps per DoF	–	–	2 000 000	200 000	2 000 000	–
	SA steps per DoF per cycle	–	–	1	1	1	–
	Subdomain size	–	–	25	25	25	–

Table 3.13: Number of SA steps and SA steps per DoF per sweep used for numerical results in right-hand panel of Figure 3.7.

SA steps / DoF	2×2	3×3	4×4	5×5	6×6
5000	20	10	5	1	1
200 000	10	2	50	20	2
2 000 000	–	–	50	1	1

Table 3.14: Number of SA steps per DoF per sweep used for numerical results in right-hand panel of Figure 3.8.

# of nodes per aggregate	2	5	8	11	14	16	25	36
FD:								
SA steps per DoF	200 000	7500	100 000	200 000	200 000	1 000 000	500 000	2 000 000
SA steps per DoF per cycle	10	10	20	5	50	25	1	2
FE:								
SA steps per DoF	7500	200 000	2 000 000	200 000	200 000	200 000	2 000 000	2 000 000
SA steps per DoF per cycle	5	100	20	50	1	1	1	1

Table 3.15: Number of SA steps and SA steps per DoF per sweep used for numerical results in Figure 3.10.

Partitioning		FD Discretization			FE Discretization		
		32×32	64×64	128×128	32×32	64×64	128×128
Geometric	SA steps per DoF	200 000	200 000	200 000	200 000	200 000	200 000
	SA steps per DoF per cycle	1	1	1	2	1	1
	Subdomain size	6×6	6×6	6×6	5×5	5×5	5×5
Algebraic	SA steps per DoF	200 000	200 000	200 000	2 000 000	200 000	2 000 000
	SA steps per DoF per cycle	1	1	1	1	1	2
	Subdomain size	16	36	36	20	36	36

Table 3.16: Number of SA steps per DoF and SA steps per DoF per sweep and subdomain sizes used for two-level numerical results in Table 3.5.

	Table 3.7			Table 3.8		
	1433	5617	22241	798	3109	12273
SA steps per DoF	1 000 000	2 000 000	200 000	500 000	2 000 000	200 000
SA steps per DoF per cycle	5	2	1	1	2	1
Subdomain size	20	36	36	36	36	36

Table 3.17: Number of SA steps per DoF and SA steps per DoF per sweep and subdomain sizes used for two-level numerical results in Tables 3.7 and 3.8.

Chapter 4

Generalizing Reduction-Based Algebraic Multigrid

Abstract

[1] Algebraic Multigrid (AMG) methods are often robust and effective solvers for solving the large and sparse linear systems that arise from discretized PDEs and other problems, relying on heuristic graph algorithms to achieve their performance. Reduction-based AMG (AMGr) algorithms attempt to formalize these heuristics by providing two-level convergence bounds that depend concretely on properties of the partitioning of the given matrix into its fine- and coarse-grid degrees of freedom. MacLachlan and Saad (SISC 2007) proved that the AMGr method yields provably robust two-level convergence for symmetric and positive-definite matrices that are diagonally dominant, with a convergence factor bounded as a function of a coarsening parameter. However, when applying AMGr algorithms to matrices that are not diagonally dominant, not only do the convergence factor bounds not hold, but measured performance is notably degraded. Here, we present modifications to the classical AMGr algorithm that improve its performance on matrices that are not diagonally dominant, making use of strength of connection, sparse approximate inverse (SPAI) techniques, and interpolation truncation and rescaling, to improve robustness while maintaining control of the algorithmic costs. We present numerical results demonstrating the robustness

[1] This work is submitted as "Generalizing Reduction-Based Algebraic Multigrid" by Tareq Zaman, Nicolas Nytko, Ali Taghibakhshi, Scott MacLachlan, Luke Olson, and Matthew West, to Numerical Linear Algebra with Applications, 2022.

of this approach for both classical isotropic diffusion problems and for non-diagonally dominant systems coming from anisotropic diffusion.

Keyword: Algebraic Multigrid, Reduction-based Multigrid, Sparse Approximate Inverse.

4.1 Introduction

Partial differential equations (PDEs) arise naturally as mathematical models of physical systems in many fields of science and engineering. As analytical techniques for their solution are limited to simple equations and geometries, numerical approximation of solutions via discretization techniques is ubiquitous. Standard discretizations necessitate the solution of large linear and nonlinear systems of equations which, in turn, requires fast and efficient algorithms. Among the techniques most commonly used for discretized elliptic equations are multigrid (MG) methods, known for their efficient and robust solution of a wide range of problems. Geometric multigrid (GMG) methods are often most efficient, but require detailed knowledge of the problem to be solved, its discretization, and regular structure of the underlying mesh hierarchy. In contrast, algebraic multigrid (AMG) methods can be effectively applied to problems on unstructured grids, or with highly variable (or discontinuous) coefficients. While the idea of AMG was first proposed over 40 years ago [12, 13, 52, 54], understanding and improving the convergence of AMG remains an active area of research.

Classical (Ruge-Stüben) AMG has become a workhorse algorithm in scientific computing, particularly due to high-quality implementations available in standard packages [2, 28, 36]. While it provides an efficient and robust solution algorithm for a wide class of diffusion problems, its reliance on heuristics and algorithmic parameters (that can be difficult to tune) is often seen as a difficult hurdle to overcome, particularly for critical applications. As a result, significant effort has been invested in recent years in the development of AMG approaches with rigorous convergence bounds. Within this area are several approaches based on the pairwise aggregation methodology [15, 18, 47, 48, 49] that offers guaranteed convergence for problems such as the graph Laplacian. An alternative approach builds on the reduction-based multigrid methodology first proposed by Ries, Trottenberg, and Winter [51]. The reduction-based algebraic multigrid (AMGr) methodology introduced by MacLachlan et al. [40] uses algebraic properties of the linear system to determine reduction-like grid-transfer

operators and relaxation that again leads to guaranteed convergence rates.

The basic principle of reduction-based multigrid follows from classical cyclic reduction algorithms [55] that reduce the cost of the direct solution of linear systems, $A\boldsymbol{x} = \boldsymbol{b}$, by partitioning the degrees of freedom into two sets that we denote by F and C (in typical multigrid notation). The key feature of cyclic reduction is that this partitioning should be done in a way so that the submatrix of A over the set F, denoted A_{FF}, is a diagonal matrix. (Equivalently, the set F is an *independent set* in the graph associated with the sparse matrix, A.) Multigrid reduction [51] and AMGr [40] generalize this by allowing A_{FF} to be non-diagonal (F is not required to be an independent set), but only spectrally equivalent to a diagonal matrix. A key question left unanswered in this work is how to generate such partitions. MacLachlan and Saad [39] show that the task of generating the largest possible F set is an integer linear programming problem and, consequently, proposed a greedy algorithm for the partitioning with linear complexity. Notably, they proved that if A is symmetric, positive definite, and diagonally dominant, then there is a two-grid AMGr method with a guaranteed convergence bound. Several theoretical and practical improvements to the theory and algorithms of MacLachlan and Saad [39] have been proposed, with improved AMGr algorithms and convergence bounds [17, 32] and improved coarsening algorithms [56, 60]. Nonsymmetric variants have also been considered [38, 41, 42]. Further research into multigrid reduction (but not AMGr) has also been commonplace in recent years, with the emergence of multigrid-reduction-in-time [29] (MGRIT) methodologies for the solution of time-dependent PDEs.

Despite these developments, attaining effective AMGr convergence has remained essentially limited to systems with matrices that are either close to diagonally dominant or can be reordered to be close to lower triangular. Unfortunately, these limitations are significant, and preclude applying AMGr to many important and interesting classes of problems, including common AMG test cases, such as anisotropic diffusion equations, or problems discretized on anisotropic meshes. Poor performance on anisotropic problems, in particular, is reported in existing AMGr results [39, 60] that has only been overcome with expensive and impractical fixes, such as moving substantial numbers of points from F to C, in order to construct suitably improved interpolation operators within the AMGr framework. In this paper, we revisit the basic AMGr framework, with the goal of overcoming the barriers to achieving acceptable convergence for anisotropic problems. To do this, we look for more practical algorithmic choices within an AMGr-style algorithm. Specifically, we consider four

ingredients for improving AMGr performance:

1. C-relaxation; while original AMGr focused on F-relaxation, we show that using C-relaxation (as considered in other settings [17, 29]) can be particularly helpful for these problems;

2. Sparse Approximate Inverses (SPAI); while the original AMGr papers focused on approximating A_{FF}^{-1} by a diagonal matrix, we consider using the SPAI algorithm [7, 34, 37] to offer better approximation of A_{FF}^{-1} while still maintaining sparsity;

3. Strength of Connection; while the original AMGr algorithms do not rely on the classical AMG [13, 52] notion of "strong connections" to filter small entries from the matrix, we find that such filtering is critical to success for anisotropic problems, where large "wrong-sign" off-diagonal entries appear, but cannot be productively used in the relaxation or coarse-grid correction processes; and

4. Interpolation truncation; the combination of techniques described above leads to effective AMGr-style solvers for a wider range of problems, but with higher algorithmic complexity than is needed; thus, we use the technique of interpolation truncation [25, 26, 53] to control these costs.

Numerical results demonstrate that these techniques can be used in combination to improve the performance of AMGr for both isotropic and anisotropic diffusion problems.

Sparse approximate inverses are one of a class of algorithms that define preconditioners for $A\boldsymbol{x} = \boldsymbol{b}$ by prescribing a form for matrix M, then minimizing some norm of $I - MA$ (or $I - AM$). Originally proposed and investigated by Benson and Frederickson [3, 5, 31], recent investigations include factorized sparse approximate inverses [37] (FSAI), which aim to compute a sparse approximation to the Cholesky factorization of SPD matrix A, and SPAI techniques [7, 24, 34] that directly compute sparse approximations to A^{-1}. The use of SPAI techniques in multigrid methods dates back almost to their initial introduction [4, 31], primarily to replace the use of standard relaxation schemes, such as the weighted Jacobi and Gauss-Seidel iterations. The use of SPAI techniques for relaxation within both geometric and algebraic multigrid has been considered more recently in several ways [10, 21, 22, 23, 30, 33, 59]. Similar ideas have been used in other contexts, to build interpolation operators [46, 58] or improve coarse-grid operators [11]. The work in this paper is closest to the ideas presented

by Bollhöfer [9], where SPAI was used to determine both the relaxation scheme and the interpolation operator, although the remaining details of the scheme are quite different. We also note similarity to the work of Meurant [44, 45], where entries from the AINV [8] preconditioner were directly used to determine interpolation alongside AINV for relaxation.

The remainder of this paper is organized as follows. In Section 4.2, we give an introduction to algebraic multigrid, with a particular focus on reduction-based AMG (AMGr). We highlight, in Section 4.2.3, that AMGr as it exists has significant difficulties for anisotropic diffusion equations, motivating the work that follows. A key component of the algorithms considered here is the Sparse Approximate Inverse methodology of Grote and Huckle [34], we review this as well in Section 4.3. The main contribution of this paper is the generalized AMGr algorithm developed in Section 4.4. Supporting numerical results are presented in Section 4.5, followed by conclusions in Section 4.6.

4.2 Algebraic multigrid

Multigrid methods are based on the principle of complementarity, using fine-grid relaxation and coarse-grid correction to efficiently damp all errors in the approximation of solutions to linear systems $A\boldsymbol{x} = \boldsymbol{b}$. Geometric multigrid methods (GMG) fix a multigrid hierarchy by directly discretizing the PDE on a series of meshes defined by the problem geometry, and by adapting the relaxation scheme to complement the coarse-grid correction process defined in this way. Algebraic multigrid methods, in contrast, do not rely on explicit knowledge of the geometry nor the PDE, instead determining the coarse levels of the multigrid hierarchy in a setup phase that precedes the solution phase of the multigrid algorithm. In the setup, the set of degrees of freedom (or points) on the finest grid, Ω, is partitioned into disjoint sets, $\Omega = C \cup F$ (with $C \cap F = \emptyset$). The degrees of freedom in the set C constitute the points on the second level. Along with this partitioning, an interpolation operator, P, is constructed to map vectors from C onto Ω; similarly a restriction operator is defined, $R = P^T$ (in the symmetric case, as considered here), to map vectors from Ω onto C. With this, the Galerkin coarse-grid operator, $A_C = P^T A P$, is formed and the process continues recursively on A_C and C. The hierarchy is constructed until the number of nodes on a coarse grid is sufficiently small that direct factorization of A_C is feasible. The algorithm for a two-level setup phase is shown in Algorithm 4.1. Once the setup phase is

Algorithm 4.1 AMG Setup Phase

```
1: function AMG-TWO-LEVEL-SETUP(A)
2:     C, F ← split the degrees of freedom into coarse and fine nodes
3:     P ← form interpolation operator
4:     A_c ← P^T A P
5:     return (P, A_c)
```

Algorithm 4.2 AMG Solution Phase

```
1: function AMG-TWO-LEVEL-V-CYCLE(A, b, x, P)
2:     for j ← 1, ..., ν_1 do          ▷ Run ν_1 sweeps of pre-relaxation
3:         x ← relax on x
4:     r_C ← P^T (b − A x)
5:     e_C ← solution of A_C e_C = r_C          ▷ Solve the coarse-level problem
6:     x ← x + P e_C
7:     for j ← 1, ..., ν_2 do          ▷ Run ν_2 sweeps of post-relaxation
8:         x ← relax on x
9:     return x
```

completed, the solution phase solves the original system of equations using a standard multigrid cycling algorithm. A two-level algorithm is shown in Algorithm 4.2. Multilevel generalizations come from recursively solving $A_C \boldsymbol{e}_C = \boldsymbol{r}_C$ using the two-grid methodology, either once per level (leading to a V-cycle) or multiple two-grid sweeps per level (leading, for example, to the W-cycle). AMG methods are generally distinguished by how they define C from Ω, and how they define P from C and A. Below, we review the classical AMG algorithm, as presented by Ruge and Stüben [52], and the reduction-based AMG algorithm of MacLachlan et al. [40].

4.2.1 Classical (Ruge-Stüben) AMG

Classical AMG makes use of the notion of *strong connections* in the graph corresponding to matrix A to define the coarse-fine partitioning. In the original algorithm, point i is said to be strongly connected to point j if $-A_{ij} \geq \gamma \max_{k \neq i}(-A_{ik})$ for some $\gamma \in (0, 1]$ (where the negative signs reflect the expectation that A be an M-matrix, or one where positive off-diagonal entries are not substantial). Many other definitions of strong connections have been considered in the literature, including the symmetric

strength measure commonly used in smoothed aggregation AMG, where the connection between i and j is said to be strong if $|A_{ij}| \geq \gamma\sqrt{A_{ii}A_{jj}}$. Local approximations of the inverse of A and algebraically smooth error have also been used to define strong connections [16, 50], as has the concept of algebraic distances [14]. In all cases, the strength measure is used to "filter" the entries in A, and then the set C is defined by choosing a maximal independent set over the graph of strong connections. This can be characterized, for example, by the properties that each point in F is strongly dependent on at least one C-point, but no two C-points can be strongly dependent on one another.

The standard interpolation operator in classical AMG is formulated based on the assumption that algebraically smooth errors have small residuals after relaxation. Hence, the residual equation after relaxation can be written as $A\boldsymbol{e} \approx \mathbf{0}$. For the i-th degree of freedom, $i \in F$, this gives

$$A_{ii}e_i \approx -\sum_{k\in C_i} A_{ik}e_k - \sum_{j\in F_i^s} A_{ij}e_j - \sum_{\ell\in F_i^w} A_{i\ell}e_\ell, \tag{4.1}$$

where C_i, F_i^s, and F_i^w are the C-neighbors, strongly connected F-neighbors, and weakly connected F-neighbors of the i-th point, respectively. The standard interpolation operator is then derived by assuming that $e_\ell \approx e_i$ for weakly-connected neighbors, while

$$e_j \approx \frac{\sum_{k\in C_i} A_{jk}e_k}{\sum_{m\in C_i} A_{jm}},$$

for points $j \in F_i^s$. Substituting these into (4.1) and solving for e_i yields the interpolation formula

$$e_i = \sum_{k\in C_i} \frac{-A_{ik} - \sum_{j\in F_i^s} \frac{A_{ij}A_{jk}}{\sum_{m\in C_i} A_{jm}}}{A_{ii} + \sum_{\ell\in F_i^w} A_{i\ell}} e_k. \tag{4.2}$$

This defines the interpolation formula for all points $i \in F \subset \Omega$, assuming that e_k is known for all $k \in C_i$. For points $k \in C$, we use direct injection of values from coarse-grid points to their fine-grid counterparts in Ω.

4.2.2 Reduction-based algebraic multigrid (AMGr)

Cyclic reduction [55] was originally proposed as a direct solver for certain linear systems that arose from finite-difference discretization of simple PDEs. Assuming that the degrees of freedom are already partitioned into coarse and fine nodes, the

linear system $A\boldsymbol{x} = \boldsymbol{b}$ is reordered to have F degrees of freedom followed by C degrees of freedom, writing

$$A = \begin{bmatrix} A_{FF} & -A_{FC} \\ -A_{CF} & A_{CC} \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_F \\ \boldsymbol{x}_C \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}_F \\ \boldsymbol{b}_C \end{bmatrix}. \tag{4.3}$$

An *exact* algorithm for the solution of $A\boldsymbol{x} = \boldsymbol{b}$ in this partitioned form is given by

1. $\boldsymbol{y}_F = A_{FF}^{-1}\boldsymbol{b}_F$,
2. Solve $\left(A_{CC} - A_{CF}A_{FF}^{-1}A_{FC}\right)\boldsymbol{x}_C = \boldsymbol{b}_C + A_{CF}\boldsymbol{y}_F$,
3. $\boldsymbol{x}_F = \boldsymbol{y}_F + A_{FF}^{-1}A_{FC}\boldsymbol{x}_C$.

This can be turned into an iterative method for solving $A\boldsymbol{x} = \boldsymbol{b}$ in the usual way, replacing the right-hand side vector, $\boldsymbol{b}$, by the evolving residual, leading to reduction-based multigrid [51]. In this form, we compute updates to the current approximation, $\boldsymbol{x}^{(k)}$, as

1. $\boldsymbol{x}_F^{(k+1/2)} = \boldsymbol{x}_F^{(k)} + A_{FF}^{-1}\left(\boldsymbol{b}_F - A_{FF}\boldsymbol{x}_F^{(k)} + A_{FC}\boldsymbol{x}_C^{(k)}\right)$,
2. Solve $\left(A_{CC} - A_{CF}A_{FF}^{-1}A_{FC}\right)\boldsymbol{y}_C = \boldsymbol{b}_C + A_{CF}\boldsymbol{x}_F^{(k+1/2)} - A_{CC}\boldsymbol{x}_C^{(k)}$,
3. $\boldsymbol{x}_C^{(k+1)} = \boldsymbol{x}_C^{(k)} + \boldsymbol{y}_C$,
4. $\boldsymbol{x}_F^{(k+1)} = \boldsymbol{x}_F^{(k+1/2)} + A_{FF}^{-1}A_{FC}\boldsymbol{y}_C$.

We note that, as given above, this is still an exact algorithm, as $\boldsymbol{x}^{(1)} = \boldsymbol{x} = A^{-1}\boldsymbol{b}$ for any initial guess, $\boldsymbol{x}^{(0)}$. To make an iterative method from this skeleton, we introduce approximations of A_{FF}^{-1} in three places in the above algorithm, namely

$$\tilde{A}_{FF}^{-1} \approx A_{FF}^{-1} \qquad \tilde{A}_C \approx A_{CC} - A_{CF}A_{FF}^{-1}A_{FC} \qquad W_{FC} \approx A_{FF}^{-1}A_{FC}, \tag{4.4}$$

leading to the iteration:

1. $\boldsymbol{x}_F^{(k+1/2)} = \boldsymbol{x}_F^{(k)} + \tilde{A}_{FF}^{-1}\left(\boldsymbol{b}_F - A_{FF}\boldsymbol{x}_F^{(k)} + A_{FC}\boldsymbol{x}_C^{(k)}\right)$,
2. Solve $\tilde{A}_C\boldsymbol{y}_C = \boldsymbol{b}_C + A_{CF}\boldsymbol{x}_F^{(k+1/2)} - A_{CC}\boldsymbol{x}_C^{(k)}$,
3. $\boldsymbol{x}_C^{(k+1)} = \boldsymbol{x}_C^{(k)} + \boldsymbol{y}_C$,
4. $\boldsymbol{x}_F^{(k+1)} = \boldsymbol{x}_F^{(k+1/2)} + W_{FC}\boldsymbol{y}_C$.

Viewing this as a two-grid algorithm, we recognize the first step as special form of relaxation, known as F-relaxation, where the approximation to A_{FF}^{-1} is accomplished via a standard weighted Jacobi or Gauss-Seidel iteration. The second step then represents a coarse-grid solve, where the residual is restricted to the coarse-grid by injection, and the correction, $\boldsymbol{y}_C$, is computed using an approximation, $\tilde{A}_C$, of the true Schur complement, $A_{CC} - A_{CF}A_{FF}^{-1}A_{FC}$. The final two steps represent the interpolation of the correction, writing the interpolation operator $P = \left[\begin{smallmatrix} W_{FC} \\ I \end{smallmatrix}\right]$. This can be viewed as an approximation of the *ideal* interpolation operator [27], $W_{FC} \approx A_{FF}^{-1}A_{FC}$. Notably, this algorithm differs from standard multigrid cycling in several ways, including the fixed use of injection for the restriction of the residual to the coarse grid, and the lack of post-relaxation sweeps.

As written above, there is little guidance in how to choose the three approximations in (4.4). MacLachlan et al. [40] address this in their development of the reduction-based AMG (AMGr) algorithm, connecting convergence of the two-grid scheme with properties of A_{FF}. In particular, it is assumed that A_{FF} can be approximated by known matrix D_{FF} for which computing the action of D_{FF}^{-1} on a vector is computationally feasible. From this, Theorem 4.1 holds, using the notation that matrices $A \succeq B$ when $\boldsymbol{x}^T A\boldsymbol{x} \geq \boldsymbol{x}^T B\boldsymbol{x}$ for all vectors $\boldsymbol{x}$.

Theorem 4.1. *[40] Consider the symmetric and positive-definite matrix* $A = \left[\begin{smallmatrix} A_{FF} & -A_{FC} \\ -A_{FC}^T & A_{CC} \end{smallmatrix}\right]$ *such that* $A_{FF} = D_{FF} + \mathcal{E}$, *with* D_{FF} *symmetric,* $0 \preceq \mathcal{E} \preceq \epsilon D_{FF}$ *for some* $\epsilon \geq 0$, *and* $\left[\begin{smallmatrix} D_{FF} & -A_{FC} \\ -A_{FC}^T & A_{CC} \end{smallmatrix}\right] \succeq 0$. *Define relaxation with error-propagation operator* $R = \left(I - \sigma \left[\begin{smallmatrix} D_{FF}^{-1} & 0 \\ 0 & 0 \end{smallmatrix}\right] A\right)$ *for* $\sigma = 2/(2+\epsilon)$, *interpolation* $P = \left[\begin{smallmatrix} D_{FF}^{-1}A_{FC} \\ I \end{smallmatrix}\right]$, *and coarse-level correction with error-propagation operator* $T = I - P(P^TAP)^{-1}P^TA$. *Then the multigrid cycle with* ν *pre-relaxation sweeps, coarse-level correction, and* ν *post-relaxation sweeps has error propagation operator* $MG_2 = R^\nu \cdot T \cdot R^\nu$ *which satisfies*

$$||MG_2||_A \leq \left(\frac{\epsilon}{1+\epsilon}\left(1 + \left(\frac{\epsilon^{2\nu-1}}{(2+\epsilon)^{2\nu}}\right)\right)\right)^{1/2} < 1. \tag{4.5}$$

Several generalizations of both the theory and practice of reduction-based multigrid methods have since been developed. A generalization to non-symmetric M-matrices was proposed and analyzed by Mense and Nabben [43], using the tools of weak regular splitting [57]. For symmetric and positive definite problems, Brannick et al. [17] study the introduction of more general relaxation schemes, as well as the use of different approximations of A_{FF} for interpolation and relaxation. Gossler and

Nabben [32] examine generalization of AMGr to the use of Chebyshev polynomial acceleration of multiple relaxation sweeps. For strongly non-symmetric systems, Manteuffel et al. [41, 42] have proposed similar approaches using so-called approximate ideal restriction (AIR) techniques, that offer excellent performance for advection-dominated problems. None of the above schemes, however, address the poor performance observed in AMGr-type methods for anisotropic problems, which is the motivation for the present work.

4.2.2.1 Coarsening in AMGr

Theorem 4.1 establishes existence of an interpolation operator, P, provided that the matrix A can be partitioned into $\begin{bmatrix} A_{FF} & -A_{FC} \\ -A_{FC}^T & A_{CC} \end{bmatrix}$ and an approximation, D_{FF}, of A_{FF} can be made that satisfies the assumptions in the theorem. While this is insightful, it does not address the fundamental question of how to generate a partitioning for which the assumptions hold with small parameter ϵ. To answer this question, MacLachlan and Saad [39] propose to partition the rows and columns of A in order to ensure the diagonal dominance of A_{FF}, allowing D_{FF} to be chosen as a diagonal matrix. In particular, for each row, i, the diagonal dominance of row i over the F points is quantified by

$$\eta_i = \frac{|A_{ii}|}{\sum_{j \in F} |A_{ij}|}.$$

Then, A_{FF} is said to be η-diagonally dominant if $\eta_i \geq \eta$ for all $i \in F$, for some $\eta > 1/2$ that measures the diagonal dominance of A_{FF}. If A_{FF} is η-diagonally dominant, then the diagonal matrix, D_{FF}, with $(D_{FF})_{ii} = (2 - \frac{1}{\eta})A_{ii}$ for all $i \in F$ yields $0 \preceq \mathcal{E} \preceq \frac{2-2\eta}{2\eta-1} D_{FF}$, giving an η-dependent convergence bound if the other assumptions of Theorem 4.1 are satisfied. Furthermore, if A is symmetric, positive-definite, and diagonally dominant, then this condition guarantees that all conditions of Theorem 4.1 are satisfied.

In addition to establishing this connection between the diagonal dominance parameter η and the convergence parameter, ϵ, MacLachlan and Saad [39] pose the partitioning algorithm as an optimization problem: for a given $\eta > 1/2$, find the largest F-set such that $\eta_i \geq \eta$ for every $i \in F$. This can be written as

$$\begin{aligned} &\max_{F \subset \Omega} |F|, \\ &\text{subject to } |A_{ii}| \geq \eta \sum_{j \in F} |A_{ij}|, \ \forall i \in F. \end{aligned} \tag{4.6}$$

MacLachlan and Saad [39] show that finding the optimal solution to (4.6) is NP-complete and, consequently, propose a greedy algorithm to approximately solve the optimization problem. The greedy algorithm acts iteratively, adding points to the C-set one at a time, and moving any points that are guaranteed to satisfy the inequality constraint in (4.6) into the F-set, until a full partition is computed.

While the greedy coarsening algorithm was demonstrated to be effective in some settings, Zaman et al. [60] demonstrate that there are also cases where the resulting optimality gap can be significant. To address this, they propose to apply simulated annealing to the optimization problem in (4.6). In this approach, the set of points, Ω, is first partitioned into non-overlapping subdomains, and a local partitioning is computed on each subdomain, starting from the assumption that all points are to be made C points. Using a Gauss-Seidel-like approach, the subdomains are processed successively, running a number of annealing steps on points within the subdomain, randomly swapping some C points to be F points or vice-versa, and accepting the move (with some probability) if it improves the overall fitness (quantified by the number of F points that satisfy the constraint in (4.6)). Zaman et al. [60] show that this approach can produce substantially better partitionings than the greedy approach, albeit at the greatly increased cost of many simulated annealing steps. Similar work by Taghibakhshi et al. [56] uses reinforcement learning to solve the same problem at a lower cost. In both of these papers, while "good" solutions are found to the optimization problem in (4.6), poor performance is observed for anisotropic diffusion problems, as these problems do not satisfy the diagonal dominance condition required by the theory of MacLachlan and Saad [39] in the assumptions of Theorem 4.1.

4.2.3 Failure of AMGr for anisotropic diffusion

To demonstrate the convergence problems, we consider applying AMGr (following the prescription of MacLachlan and Saad [39]) to the solution of the two-dimensional anisotropic diffusion problem,

$$-\nabla \cdot \boldsymbol{K}(x,y)\nabla u(x,y) = b(x,y) \tag{4.7}$$

in the domain $[0,1]\times[0,1]$ with Dirichlet boundary conditions. We choose the tensor coefficient $\boldsymbol{K}(x,y) = QHQ^T$, where

$$Q = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}, \qquad H = \begin{bmatrix} 10^{-6} & 0 \\ 0 & 1 \end{bmatrix}, \tag{4.8}$$

Table 4.1: Performance of two-level AMGr for the anisotropic diffusion problem.

θ	Discretization	Eigenvalues of $D_{FF}^{-1}A_{FF}$ min	max	$\begin{bmatrix} D_{FF} & -A_{FC} \\ -A_{FC}^T & A_{CC} \end{bmatrix}$	Convergence factor ρ	Complexities C_{grid}	C_{op}
0	FD	1.00	3.00	positive definite	0.75	1.33	1.49
	FE	1.00	6.83	indefinite	0.98	1.49	2.14
$\pi/6$	FD	1.04	5.59	indefinite	0.96	1.42	1.87
	FE	1.00	7.09	indefinite	0.97	1.36	1.72
$\pi/4$	FD	1.21	6.73	indefinite	0.96	1.39	1.79
	FE	1.12	5.80	indefinite	0.96	1.33	1.59

and θ specifies the direction of anisotropy in the problem. For $\theta = 0$ this gives the grid-aligned anisotropic equation $-10^{-6}u_{xx} - u_{yy} = b$, while $0 < \theta < \pi/2$ gives a non-grid-aligned diffusion tensor. Table 4.1 presents convergence results for the standard finite-difference and bilinear finite element discretizations of this problem, with $\theta = 0$, $\pi/6$, and $\pi/4$. For simplicity, we present results for a uniform 32×32 grid, although similar results are observed for larger meshes. Here, and in all results that follow, we measure convergence by solving the homogeneous problem, $A\boldsymbol{x} = \boldsymbol{0}$, with a randomly chosen initial guess for $\boldsymbol{x}$. Writing $\boldsymbol{e}^{(k)}$ as the error in the k^{th} approximation to $\boldsymbol{x} = \boldsymbol{0}$, we estimate the asymptotic convergence factor by running 50 (stationary) multigrid iterations, then estimating $\rho \approx \left(\|\boldsymbol{e}^{(50)}\|_A / \|\boldsymbol{e}^{(10)}\|_A\right)^{1/40}$, averaging convergence over the final 40 iterations. We note that, in all cases, the coarsening algorithm (simulated annealing, in this case) generates a partitioning such that the matrix A_{FF} is well-approximated by a diagonal matrix, D_{FF}; however, only in the case of the finite-difference discretization of the grid-aligned diffusion equation (when the discretization matrix, A, *is* diagonally dominant), does the required semidefiniteness of $\begin{bmatrix} D_{FF} & -A_{FC} \\ -A_{FC}^T & A_{CC} \end{bmatrix}$ hold. This correlates strongly with the resulting measured asymptotic convergence factor for the method. Both MacLachlan and Saad [39] and Zaman et al. [60] consider remedies for this behavior, such as augmenting the C set in a style similar to classical AMG; while this improves the overall convergence of the method, it also leads to greatly increased grid and operator complexities, making it an unsatisfactory solution.

Here, and in all tables that follow, we use color-coding to indicate quality of the results shown. For measured convergence factors, we denote a "good" convergence factor to be below 0.4 (indicated in green), while a "bad" convergence factor, above

0.8, is shown in red (with values in between, $0.4 < \rho < 0.8$, shown in black text). For two-grid operator complexity, we highlight results in green if the complexity is below 1.5, in red if it is above 2.5, and in orange for values between 2.0 and 2.5. As operator complexity, in particular, is expected to grow with the number of levels in the hierarchy, we use similar highlighting with different thresholds for three-grid and multigrid operator complexity, showing results in green if it is below 2.0, red if it is above 3.0, and orange for values between 2.5 and 3.0. As the results that follow show relatively little variation in AMG grid complexity, we choose not to highlight values for this measure. Definitions of these complexities are found below, in Section 4.4.1.

4.3 Sparse Approximate Inverse (SPAI) methods

While originally proposed in the 1970's by Benson and Frederickson [3, 31], SPAI techniques were more systematically developed and studied in the 1990's (and subsequently) by a number of authors [6, 7, 24, 34, 37]. The general idea of SPAI techniques is to compute a matrix, M, to minimize some norm of $I - MA$ or $I - AM$, with constraints on the sparsity of M. These constraints may be fixed (e.g., some fixed set of elements of M is allowed to be nonzero), or may be adaptively determined by trying to best minimize the chosen norm within some limitations on either the total number of nonzero entries in M or the row/column-wise number of nonzero elements. Here, we focus on the variant of the SPAI algorithm proposed by Hawkins and Chen [35], in which the Frobenius norm of $B - AM$ is minimized for a given matrix, B, over a fixed nonzero pattern for each column. If $B = I$, then this reduces to a simplified version of the SPAI algorithm of Grote and Huckle [34], omitting their adaptive calculation for increasing the nonzero pattern for each column.

Algorithm 4.3 presents the SPAI algorithm, where the inputs are given by matrices A and B, and a nonzero sparsity pattern, $\mathcal{S}$, for the sparse approximate inverse M. The algorithm loops independently over each column, j, in M. In the initialization stage of the algorithm (Lines 3 through 7), the rows, $\mathcal{J}$, in the initial sparsity pattern of $\mathcal{S}$ for column j are extracted, as is the set of rows, $\mathcal{I}$, of A for which matrix A has a nonzero entry in a column in $\mathcal{J}$, defining a submatrix, $\bar{A}$, of A that is used to initialize column j of M. Two auxiliary vectors are also formed, corresponding to the full j^{th} column of B, denoted $\boldsymbol{b}$, and its restriction to the rows of $\bar{A}$, denoted $\bar{\boldsymbol{b}}$. Column j of M then comes from using the QR decomposition of $\bar{A}$ to solve the unconstrained minimization problem of minimizing $\left\|\bar{\boldsymbol{b}} - \bar{A}\bar{\boldsymbol{m}}\right\|_2$, noting that $\bar{A}$ is expected, by its

construction, to have more rows than columns, so that this is not expected to yield a zero residual. The computed solution, $\bar{\boldsymbol{m}}$, is injected into a full vector, $\boldsymbol{m}$, which becomes the j^{th} column of M.

Algorithm 4.3 SPAI algorithm

```
1:  function SPAI(A, B, S)
2:      for j ← 1, ..., n_col do                    ▷ n_col is number of columns in A
3:          J ← {i | (i, j) ∈ S}
4:          I ← set of indices of nonzero rows of A(:, J)
5:          b ← B(:, j)
6:          Ā ← A(I, J)
7:          b̄ ← B(I, j)
8:          Compute QR decomposition of Ā
9:          m̄ ← argmin_m̄ ||b̄ − Ām̄||_2          ▷ Unconstrained least-squares via QR
10:         m ← m̄ with inserted zeros
11:         M(:, j) ← m
12:     return M
```

Sparse approximate inverse algorithms similar to Algorithm 4.3 have been investigated for use in both relaxation and interpolation in several settings in the past. However, this usage has generally been in defining approximations to the inverse of A in its entirety, while we look at the possible use of SPAI techniques through the lens of the AMGr methodology. In what follows, we will make use of SPAI in three ways:

1. In F-relaxation where, given a proxy matrix, $\hat{A}_{FF}$, for A_{FF} (possibly equal to A_{FF}), and $B = I_{FF}$, D_{FF}^{inv} is constructed as the SPAI approximation to $\hat{A}_{FF}^{-1}$ with a fixed sparsity pattern equal to that of $\hat{A}_{FF}$;

2. In C-relaxation where, given a proxy matrix, $\hat{A}_{CC}$, for A_{CC} (possibly equal to A_{CC}), and $B = I_{CC}$, D_{CC}^{inv} is constructed as the SPAI approximation to $\hat{A}_{CC}^{-1}$ with a fixed sparsity pattern equal to that of $\hat{A}_{CC}$; and

3. In interpolation, where we use the Hawkins and Chen modification [35], to solve $\min_W \|\hat{A}_{FC} - \hat{A}_{FF}W\|_F$ for sparse approximate columns of $W = \hat{A}_{FF}^{-1}\hat{A}_{FC}$ for proxy matrices, $\hat{A}_{FF}$ and $\hat{A}_{FC}$, for A_{FF} and A_{FC}, respectively, with a fixed

nonzero pattern equal to that of $\hat{A}_{FC} + \hat{A}_{FF}\hat{A}_{FC}$.

The first two of these can be viewed as generalizations of the use of SPAI on all of A as relaxation [21, 22] to the F- and C-relaxations typically used in reduction-based AMG. The third bears similarity to Meurant's Algorithm I3 [44], where SPAI on A (either in its original ordering or reordered according to the F-C partitioning) is used to generate an approximate inverse matrix, M, from which M_{FF} is extracted to form an interpolation operator $\left[\begin{smallmatrix} M_{FF}A_{FC} \\ I \end{smallmatrix}\right]$. We note that this is akin to approximating ideal interpolation, $\left[\begin{smallmatrix} A_{FF}^{-1}A_{FC} \\ I \end{smallmatrix}\right]$ by using an approximation to $(A^{-1})_{FF}$ rather than $(A_{FF})^{-1}$. As discussed below, direct use of SPAI to approximate A_{FF}^{-1}, A_{CC}^{-1}, and $A_{FF}^{-1}A_{FC}$ in these contexts does not directly lead to effective performance, so we consider additional tools from standard AMG development to both improve convergence and lower cost.

4.4 Generalizing AMGr

Baseline results: From the results in Table 4.1 and those documented in other works [39, 56, 60], the coarse-grid correction process emerges as a primary source for the poor performance of AMGr on anisotropic problems. Using $D_{FF}^{-1}A_{FC}$ with diagonal D_{FF} for the C-to-F interpolation matrix is more restrictive than the interpolation operators used in classical multigrid, as interpolation to an F point is only allowed from directly connected C points (corresponding to the nonzero entries in A_{FC}). To test this theory, we first use SPAI to determine an interpolation operator of the form $P = \left[\begin{smallmatrix} W \\ I \end{smallmatrix}\right]$, where the sparsity pattern of W is fixed to match that of $A_{FC} + A_{FF}A_{FC}$, allowing for interpolation to a fine-grid point from both directly connected C points and C points that are directly connected to an adjacent F point. At the same time, we replace relaxation based on a diagonal stencil with relaxation using the SPAI approximation to A_{FF}^{-1} with the sparsity pattern of A_{FF}.

To test the effects of these modifications, we again consider the anisotropic diffusion equation given in (4.7), for three angles, $\theta = 0$, $\pi/6$, and $\pi/4$, with convergence factors shown in Table 4.2. To decouple the impact of these choices from that of the coarsening, we use a geometric coarse grid chosen as semi-coarsening by a factor of three in the y-direction (the direction of strong connections in these cases). We observe significant improvement in convergence in both the $\theta = 0$ and $\theta = \pi/6$ cases, in comparison to the results in Table 4.1, although performance for $\theta = \pi/4$ is much

Table 4.2: Two-level AMGr convergence factors for anisotropic FE discretization using semi-coarsening in the y direction by a factor of three. Interpolation of F-nodes uses the SPAI approximation to $A_{FF}^{-1}A_{FC}$ and the SPAI approximation to A_{FF}^{-1} is used for relaxation.

	$\theta = 0$			$\theta = \pi/6$			$\theta = \pi/4$		
Grid size	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}
16×16	0.021	1.31	1.90	0.006	1.31	1.90	0.213	1.31	1.90
32×32	0.023	1.31	2.02	0.019	1.31	2.02	0.518	1.31	2.02
64×64	0.023	1.33	2.14	0.065	1.33	2.14	0.808	1.33	2.14
128×128	0.024	1.33	2.17	0.210	1.33	2.17	0.921	1.33	2.17

worse. Yet, we also note that the complexity of these cycles is high, with two-grid operator complexities above 2.0 due to many small nonzero entries in the resulting interpolation operators that lead to large numbers of nonzero entries in the Galerkin coarse-grid operator, $P^T AP$.

Strong connections: To address the high cost encountered in the baseline, we introduce strong connections into the algorithm. A typical row in the matrix for an anisotropic diffusion operator contains both small entries and large but "wrong-sign" entries, where there are positive contributions in directions other than the strong direction of diffusion in the PDE. In response, we introduce a filtering stage, where we compute a *proxy* matrix, $\hat{A}$, for the given system matrix, A. We first compute strong connections using the classical Ruge-Stüben definition of strength of connection, defining point i to be strongly connected to point j if

$$-A_{ij} \geq \frac{1}{2} \max_{k \neq i} -A_{ik}, \tag{4.9}$$

where a strength parameter of $1/2$ is selected as is typical for anisotropic PDEs and where only negative off-diagonal entries are allowed as strong connections (also common practice).

For anisotropic diffusion equations discretized by bilinear finite elements on uniform grids, (4.9) results in two strong connections for each interior node, aligned vertically (north and south), for $\theta = 0$, two strong connections in the north-east and south-west directions for $\theta = \pi/4$, but four strong connections for $\theta = \pi/6$, including north, south, north-east, and south-west points. To preserve the row-sum that is typically needed for best AMG performance, we define $\hat{A}$ to have off-diagonal entries

Table 4.3: Two-level AMGr convergence factors for anisotropic FE discretization using semi-coarsening in the y direction by a factor of three. Interpolation of F-nodes is based on a SPAI approximation to $\hat{A}_{FF}^{-1}\hat{A}_{FC}$ and relaxation uses a SPAI approximation to $\hat{A}_{FF}^{-1}$, **where $\hat{A}$ is the "lumped" matrix of strong connections computed from A.**

	$\theta = 0$			$\theta = \pi/6$			$\theta = \pi/4$		
Grid size	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}
16×16	0.351	1.31	1.28	0.289	1.31	1.60	0.696	1.31	1.42
32×32	0.359	1.31	1.30	0.487	1.31	1.66	0.718	1.31	1.47
64×64	0.359	1.33	1.32	0.745	1.33	1.73	0.716	1.33	1.52
128×128	0.359	1.33	1.32	0.906	1.33	1.75	0.716	1.33	1.53

matching those of A for strong connections, and diagonal entries adjusted by subtracting any weak connections in each row of A from its diagonal value (so-called "lumping" of the weak connections to the diagonal, as in (4.2)).

Next, we repeat the experiments above, but define interpolation as the SPAI approximation to $\hat{A}_{FF}^{-1}\hat{A}_{FC}$ and use F-relaxation based on the SPAI approximation to $\hat{A}_{FF}^{-1}$, with results presented in Table 4.3. For $\theta = 0$ and $\theta = \pi/4$, each F-point has a single strongly-connected C neighbor and a single strongly-connected F neighbor, while each F-point for $\theta = \pi/6$ has two of each. Using semi-coarsening in the y-direction by a factor of three, this results in interpolation to each F-point from two C-points for $\theta = 0$ and $\pi/4$ and from five C-points for $\theta = \pi/6$ (where the two strongly connected F neighbors of an F-point have a total of three strongly connected C neighbors). From the table, we see that using this definition of strength results in improved and grid-independent convergence for the case of $\theta = \pi/4$, and degraded (but still grid-independent) convergence for $\theta = 0$. However, significant degradation in convergence occurs for the case of $\theta = \pi/6$. Nonetheless, the use of strong connections has greatly improved the two-grid operator complexities, particularly for the $\theta = 0$ case, where it now matches that of geometric multigrid.

Interpolation Scaling: Tables 4.2 and 4.3 underscore the potential of SPAI-based AMGr for anisotropic diffusion equations, but also highlight that acceptable and scalable convergence is not robust. From the poor convergence for $\theta = \pi/6$ in Table 4.3, we found that even if the "lumped" matrix, $\hat{A}$, used to form interpolation retains the property that rows away from boundary conditions have zero row sum

Table 4.4: Two-level AMGr convergence factors for anisotropic FE discretization using semi-coarsening in the y direction by a factor of 3. Here, F-nodes are interpolated using the SPAI approximation to $\hat{A}_{FF}^{-1}\hat{A}_{FC}$, **postprocessed to exactly interpolate the constant vector**, and the SPAI approximation to $\hat{A}_{FF}^{-1}$ is used for relaxation, where $\hat{A}$ is the "lumped" matrix of strong connections computed from A.

	$\theta = 0$			$\theta = \pi/6$			$\theta = \pi/4$		
Grid size	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}
16×16	0.751	1.31	1.28	0.641	1.31	1.60	0.765	1.31	1.42
32×32	0.776	1.31	1.30	0.640	1.31	1.66	0.751	1.31	1.47
64×64	0.772	1.33	1.32	0.649	1.33	1.73	0.744	1.33	1.52
128×128	0.772	1.33	1.32	0.656	1.33	1.75	0.741	1.33	1.53

(and that $\hat{A}$ is an M-matrix), the interpolation operator determined by SPAI does not accurately interpolate the coarse-grid constant function onto the fine-grid. This is not surprising, since SPAI computes the interpolation operator column-wise, yet interpolation to any fixed fine-grid vector is a row-wise property of matrix P.

To address the lack of constant interpolation, we post-process the interpolation generated by SPAI, using left diagonal scaling of $W \approx \hat{A}_{FF}^{-1}\hat{A}_{FC}$ so that $\mathbf{1}_F = SW\mathbf{1}_C$. This is accomplished by computing $\boldsymbol{s} = W\mathbf{1}_C$, followed by defining diagonal matrix S with entries $1/s_i$ on its diagonal. This yields the convergence factors in Table 4.4. The results offer concrete improvement over Tables 4.2 and 4.3, in that they offer scalable convergence for all three problems (without increasing grid or operator complexities). Even so, the overall convergence factors between 0.65 and 0.8 are insufficient to be considered an effective AMG solver.

One possible cause of the degraded convergence is poor interpolation near Dirichlet boundaries, where the constant vector is not an accurate indicator of the slowest-to-converge modes of relaxation. As a remedy, we use a similar scaling of interpolation, that we refer to as "improved iteration" scaling, running a set number of sweeps of (full grid) weighted Jacobi relaxation on the homogeneous problem with the constant vector as an initial guess, to produce a relaxed vector, $\boldsymbol{z}$, and followed by a similar diagonal scaling computed to ensure that $\boldsymbol{z}_F = SW\boldsymbol{z}_C$. With this modification and five sweeps of relaxation, Table 4.5 shows notable improvement in performance for both the case of $\theta = 0$ and $\theta = \pi/6$, but still disappointing (albeit grid-independent) convergence for $\theta = \pi/4$.

Table 4.5: Two-level AMGr convergence factors for anisotropic FE discretization using semi-coarsening in the y direction by a factor of 3. Here, F-nodes are interpolated using the SPAI approximation to $\hat{A}_{FF}^{-1}\hat{A}_{FC}$, **postprocessed to exactly interpolate a relaxed vector**, and the SPAI approximation to $\hat{A}_{FF}^{-1}$ is used for relaxation, where $\hat{A}$ is the "lumped" matrix of strong connections computed from A.

	$\theta = 0$			$\theta = \pi/6$			$\theta = \pi/4$		
Grid size	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}
16×16	0.379	1.31	1.28	0.177	1.31	1.60	0.720	1.31	1.42
32×32	0.382	1.31	1.30	0.197	1.31	1.66	0.719	1.31	1.47
64×64	0.379	1.33	1.32	0.256	1.33	1.73	0.717	1.33	1.52
128×128	0.377	1.33	1.32	0.284	1.33	1.75	0.717	1.33	1.53

C-relaxation: As a final modification we employ the use of C-relaxation alongside F-relaxation. As has been considered in MGRIT [29] and other contexts, replacing simple F-relaxation with sweeps of FCF-relaxation (that is, relaxation over the F-points, followed by relaxation over the C-points, then again over the F-points, with updated residual values between each sweep) is known to greatly improve multigrid performance in some settings. Results using FCF-relaxation are shown in Table 4.6, where C-relaxation is again computed with SPAI on $\hat{A}_{CC}$. We see that including C-relaxation results in a dramatic effect for $\theta = \pi/4$, reducing the convergence factor to nearly 0.1, and a notable effect for $\theta = 0$. For $\theta = \pi/6$, adding C-relaxation has little influence on convergence, noting it does not harm performance.

4.4.1 Algebraic Coarsening

Given the satisfactory results in Table 4.6, we next focus on extending these results to fully algebraic coarsening using the simulated annealing coarsening described in Section 4.2.2.1. We note that this coarsening is computationally quite expensive [60], but that it provides the best known complexities for AMGr-style methods. In Section 4.5.4, we experiment with the more feasible greedy coarsening algorithm of MacLachlan and Saad [39]. An important consideration in assessing the quality of algebraic coarsening is the resulting complexity of the multigrid algorithm, as this is no longer determined *a priori* from the fixed geometric coarsening. We use two common measures: grid and operator complexity. The AMG grid complexity, C_{grid},

Table 4.6: Two-level AMGr convergence factors for anisotropic FE discretization using semi-coarsening in the y direction by a factor of 3. Here, F-nodes are interpolated using the SPAI approximation to $\hat{A}_{FF}^{-1}\hat{A}_{FC}$, postprocessed to exactly interpolate a relaxed vector, and the SPAI approximations to $\hat{A}_{FF}^{-1}$ and $\hat{A}_{CC}^{-1}$ are used **for FCF-relaxation**, where $\hat{A}$ is the "lumped" matrix of strong connections computed from A.

	$\theta = 0$			$\theta = \pi/6$			$\theta = \pi/4$		
Grid size	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}
16×16	0.233	1.31	1.28	0.107	1.31	1.60	0.110	1.31	1.42
32×32	0.238	1.31	1.30	0.186	1.31	1.66	0.111	1.31	1.47
64×64	0.240	1.33	1.32	0.249	1.33	1.73	0.115	1.33	1.52
128×128	0.239	1.33	1.32	0.279	1.33	1.75	0.114	1.33	1.53

is the ratio of the sum of the number of DoFs on each level of multigrid hierarchy (including the finest) to that on the finest level. Similarly, the operator complexity, C_{op}, is the ratio of the sum of the number of nonzeros in the system matrices on each level of hierarchy (including the finest) to that on the finest level.

Algebraic coarsening: Tables 4.7 and 4.8 show the convergence factors and corresponding grid and operator complexities for two-grid cycles using two values of the diagonal dominance parameter in (4.6). From preliminary experiments (not reported here), we noted substantial improvement when computing the fine-coarse partitioning using $\hat{A}$ (compared with A); hence, we use $\hat{A}$ in all subsequent results.

Table 4.7 uses $\eta = 0.65$, resulting in two-level grid complexities, $C_{\text{grid}} \approx 4/3$, matching that of the geometric semi-coarsening by three used above. Using a larger parameter, $\eta = 0.75$, in Table 4.8, results in $C_{\text{grid}} \approx 3/2$, consistent with geometric semi-coarsening by a factor of two. The coarsening is visualized in Figure 4.1 for both cases and $\theta = 0$, demonstrating that, while the coarsening is still algebraic, it retains much of the geometric character of semi-coarsening. As expected, using a larger value of η leads to an improvement in convergence factors (since the coarse-grid correction is over a larger space), but also higher complexities. In particular, for the case of $\theta = 0$, we maintain complexities similar to those of geometric multigrid for these problems, with $C_{\text{op}} \approx C_{\text{grid}}$, but we also see the typical increase in AMG operator complexity faster than grid complexity for $\theta = \pi/6$ and $\pi/4$, indicating increased density in the coarse-grid operators. While the effective convergence factors, defined

Table 4.7: Two-level AMGr convergence factors and corresponding complexities for anisotropic FE discretization **using simulated annealing coarsening with** $\eta = 0.65$. F-nodes are interpolated using the SPAI approximation to $\hat{A}_{FF}^{-1}\hat{A}_{FC}$, post-processed to exactly interpolate a relaxed vector, and the SPAI approximations to $\hat{A}_{FF}^{-1}$ and $\hat{A}_{CC}^{-1}$ are used for FCF-relaxation, where $\hat{A}$ is the "lumped" matrix of strong connections computed from A.

	$\theta = 0$			$\theta = \pi/6$			$\theta = \pi/4$		
Grid size	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}
16×16	0.219	1.32	1.32	0.110	1.35	1.69	0.116	1.30	1.40
32×32	0.235	1.32	1.33	0.189	1.35	1.78	0.114	1.32	1.49
64×64	0.234	1.34	1.37	0.342	1.36	1.91	0.121	1.33	1.54
128×128	0.231	1.34	1.39	0.400	1.36	1.96	0.133	1.34	1.57

as $\rho^{1/C_{\text{op}}}$, are lower for $\eta = 0.75$ than $\eta = 0.65$, we emphasize that these are only two-grid complexities, and denser coarse-grid matrices lead to even higher three-grid complexities in the results to follow. Thus, we focus on the choice of $\eta = 0.65$ in the results below.

Interpolation truncation: To attenuate the higher complexities observed in Tables 4.7 and 4.8, we employ interpolation truncation, which is used successfully in several AMG settings [25, 26, 53]. In our approach, we first use SPAI (with a sparsity pattern of $\hat{A}_{FC} + \hat{A}_{FF}\hat{A}_{FC}$) to compute $W \approx \hat{A}_{FF}^{-1}\hat{A}_{FC}$. Then, we sweep row-wise through W, dropping entries that are less than a factor of ζ of the largest entry (by absolute value) in the row, to yield $\widehat{W}$. As a final step, we compute matrix S to match interpolation to the relaxed vector, $\boldsymbol{z}$, so that $\boldsymbol{z}_F = S\widehat{W}\boldsymbol{z}_C$. Table 4.9 shows results using $\zeta = 0.2$. For the cases of $\theta = 0$ and $\pi/4$, we see that this truncation has no real effect in comparison with results in Table 4.7. This is easily understood from the nature of strong connections in these matrices, with only two strong connections per row, so there are only two interpolation weights in a typical row, and these weights are roughly equal in size. In such cases, no effects of this truncation are expected. For $\theta = \pi/6$, we see that this truncation leads to slight improvements in operator complexities, with small effects on convergence factors. While the savings here may be minimal, we show below that interpolation truncation is an important tool in other cases, such as the isotropic Poisson problem in Section 4.5.1. We also note that further increasing the truncation parameter to $\zeta = 0.25$ starts to show significantly

Table 4.8: Two-level AMGr convergence factors and corresponding complexities for anisotropic FE discretization **using simulated annealing coarsening with** $\eta = 0.75$. F-nodes are interpolated using the SPAI approximation to $\hat{A}_{FF}^{-1}\hat{A}_{FC}$, post-processed to exactly interpolate a relaxed vector, and the SPAI approximations to $\hat{A}_{FF}^{-1}$ and $\hat{A}_{CC}^{-1}$ are used for FCF-relaxation, where $\hat{A}$ is the "lumped" matrix of strong connections computed from A.

	$\theta = 0$			$\theta = \pi/6$			$\theta = \pi/4$		
Grid size	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}
16×16	0.168	1.50	1.51	0.106	1.48	1.97	0.062	1.47	1.63
32×32	0.174	1.50	1.53	0.176	1.50	2.04	0.069	1.49	1.72
64×64	0.182	1.50	1.55	0.227	1.50	2.09	0.075	1.51	1.75
128×128	0.189	1.51	1.56	0.260	1.51	2.14	0.075	1.51	1.77

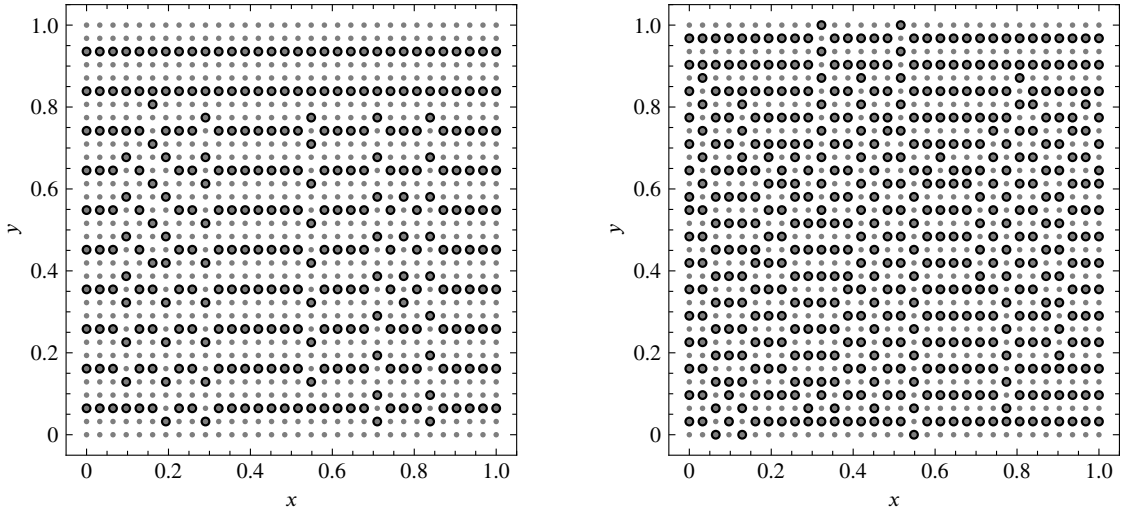


Figure 4.1: Algebraic coarse-fine partitionings for the FE discretization of anisotropic-diffusion with $\theta = 0$ on uniform 32×32 grid. At left, partitioning with $\eta = 0.65$. At right, partitioning with $\eta = 0.75$. Fine-grid DoFs (points in Ω) are denoted by filled grey dots; those that are in C are marked with black circles.

Table 4.9: Two-level AMGr convergence factors and corresponding complexities for anisotropic FE discretization using simulated annealing coarsening with $\eta = 0.65$. F-nodes are interpolated using the SPAI approximation to $\hat{A}_{FF}^{-1}\hat{A}_{FC}$, postprocessed to exactly interpolate a relaxed vector, and the SPAI approximations to $\hat{A}_{FF}^{-1}$ and $\hat{A}_{CC}^{-1}$ are used for FCF-relaxation, where $\hat{A}$ is the "lumped" matrix of strong connections computed from A. **Interpolation truncation with $\zeta = 0.2$ is used.**

	$\theta = 0$			$\theta = \pi/6$			$\theta = \pi/4$		
Grid size	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}
16×16	0.219	1.32	1.32	0.107	1.35	1.68	0.116	1.30	1.40
32×32	0.235	1.32	1.33	0.194	1.35	1.77	0.114	1.32	1.49
64×64	0.234	1.34	1.37	0.364	1.36	1.88	0.121	1.33	1.54
128×128	0.231	1.34	1.39	0.408	1.36	1.93	0.133	1.34	1.57

degraded performance for $\theta = \pi/6$, when "too many" connections in interpolation are truncated.

Tables 4.10 and 4.11 compare no-interpolation truncation, equivalent to $\zeta = 0$, to that of truncation with $\zeta = 0.2$ for three-level cycles for these problems. As expected, adding more levels to the hierarchy increases the grid and operator complexities. Using two-level geometric semi-coarsening-by-threes as a reference, we expect to see $C_{\text{grid}} \approx 1 + \frac{1}{3} + \frac{1}{9} \approx 1.44$, which we do in all cases (with a slight increase for $\theta = \pi/6$, undoubtedly due to increased density of the coarse-grid operators). We note that convergence does degrade going from two-grid to three-grid cycles, particularly for V-cycles (with convergence factors denoted by ρ_V), but also for W-cycles (with convergence factors denoted by ρ_W). Again, the effects of truncation are minimal for $\theta = 0$ and $\pi/4$, with some noticeable, but small, changes for $\pi/6$.

4.4.2 The Generalized AMGr algorithm

Before presenting more extensive numerical results, we summarize the outcome of the experiments above in algorithmic form. Algorithm 4.4 presents the AMGr setup algorithm: we compute the lumped matrix, $\hat{A}$, after finding strong connections, construct the F-C partitioning based on $\hat{A}$, then form interpolation and the Galerkin coarse-grid operator. In addition, we compute SPAI approximations to the inverses of $\hat{A}_{FF}$ and $\hat{A}_{CC}$, for use in relaxation, using $\mathcal{S}(A)$ to denote the sparsity pattern of

Table 4.10: **Three-level** AMGr convergence factors and corresponding complexities for anisotropic FE discretization using simulated annealing coarsening with $\eta = 0.65$. F-nodes are interpolated using the SPAI approximation to $\hat{A}_{FF}^{-1}\hat{A}_{FC}$, postprocessed to exactly interpolate a relaxed vector, and the SPAI approximations to $\hat{A}_{FF}^{-1}$ and $\hat{A}_{CC}^{-1}$ are used for FCF-relaxation, where $\hat{A}$ is the "lumped" matrix of strong connections computed from A.

	$\theta = 0$				$\theta = \pi/6$				$\theta = \pi/4$			
Grid size	ρ_V	ρ_W	C_{grid}	C_{op}	ρ_V	ρ_W	C_{grid}	C_{op}	ρ_V	ρ_W	C_{grid}	C_{op}
16×16	0.238	0.221	1.40	1.36	0.169	0.116	1.46	1.92	0.158	0.116	1.41	1.57
32×32	0.240	0.235	1.42	1.41	0.393	0.248	1.47	2.18	0.320	0.114	1.43	1.70
64×64	0.243	0.234	1.45	1.49	0.537	0.394	1.49	2.36	0.448	0.224	1.45	1.79
128×128	0.267	0.231	1.46	1.52	0.656	0.502	1.50	2.44	0.541	0.310	1.47	1.88

Table 4.11: **Three-level** AMGr convergence factors and corresponding complexities for anisotropic FE discretization using simulated annealing coarsening with $\eta = 0.65$. F-nodes are interpolated using the SPAI approximation to $\hat{A}_{FF}^{-1}\hat{A}_{FC}$, postprocessed to exactly interpolate a relaxed vector, and the SPAI approximations to $\hat{A}_{FF}^{-1}$ and $\hat{A}_{CC}^{-1}$ are used for FCF-relaxation, where $\hat{A}$ is the "lumped" matrix of strong connections computed from A. **Interpolation truncation with $\zeta = 0.2$ is employed.**

	$\theta = 0$				$\theta = \pi/6$				$\theta = \pi/4$			
Grid size	ρ_V	ρ_W	C_{grid}	C_{op}	ρ_V	ρ_W	C_{grid}	C_{op}	ρ_V	ρ_W	C_{grid}	C_{op}
16×16	0.238	0.220	1.40	1.36	0.175	0.113	1.46	1.85	0.153	0.116	1.41	1.57
32×32	0.248	0.235	1.42	1.41	0.359	0.231	1.47	2.06	0.323	0.114	1.44	1.70
64×64	0.243	0.234	1.45	1.49	0.556	0.411	1.49	2.24	0.437	0.215	1.46	1.79
128×128	0.262	0.231	1.46	1.52	0.670	0.542	1.50	2.31	0.534	0.298	1.47	1.87

matrix A. As in the original AMGr paper, we use weighted relaxation, with optimal weights for single-step relaxation on the F and C subproblems, computed based on eigenvalue estimates for the "preconditioned" matrices, $D_{FF}^{\text{inv}}A_{FF}$ and $D_{CC}^{\text{inv}}A_{CC}$. For the coarse-grid problem, the computed eigenvalues are very close to one in all cases, and replacing the weighted relaxation with unweighted relaxation (approximating $\sigma_C = 1$) has little effect on convergence. For the fine-grid relaxation, we find more variation in λ_{min} and λ_{max}; we explore more practical alternatives to determining re-

Algorithm 4.4 Generalized AMGr Setup Phase

function GEN-AMGR-SETUP(A, b)
  $\hat{A} \leftarrow$ lumped approximation to A after removing weak connections
  $C, F \leftarrow$ partitioning based on $\hat{A}$
  $\hat{A}_{FF}, \hat{A}_{FC}, \hat{A}_{CC} \leftarrow$ extract submatrices of $\hat{A}$ based on F and C
  $P \leftarrow$ INTERPOLATION($\hat{A}, \hat{A}_{FF}, \hat{A}_{FC}, F, C$)
  $A_C \leftarrow P^T A P$
  $D_{FF}^{\text{inv}} \leftarrow$ SPAI($\hat{A}_{FF}, I_{FF}, \mathcal{S}\left(\hat{A}_{FF}\right)$)
  $\lambda_{\min}, \lambda_{\max} \leftarrow$ minimum and maximum eigenvalues of $D_{FF}^{\text{inv}} A_{FF}$
  $\sigma_F \leftarrow 2/(\lambda_{\min} + \lambda_{\max})$
  $D_{CC}^{\text{inv}} \leftarrow$ SPAI($\hat{A}_{CC}, I_{CC}, \mathcal{S}\left(\hat{A}_{CC}\right)$)
  $\lambda_{\min}, \lambda_{\max} \leftarrow$ minimum and maximum eigenvalues of $D_{CC}^{\text{inv}} A_{CC}$
  $\sigma_C \leftarrow 2/(\lambda_{\min} + \lambda_{\max})$
  return $F, C, P, A_C, D_{FF}^{\text{inv}}, \sigma_F, D_{CC}^{\text{inv}}, \sigma_C$

laxation weights in Section 4.5.4, although note that it may also be possible to choose better weights row-wise (e.g., by using weighting similar to the ℓ^1-Jacobi relaxation method [1] or the matrix S determined for interpolation).

The interpolation operator is computed following Algorithm 4.5. First, the nonzero pattern is determined for interpolation, followed by a SPAI approximation to $\hat{A}_{FF}^{-1}\hat{A}_{FC}$ for this pattern. Small entries may be truncated in W at this stage in order to reduce complexity of the resulting cycle. After truncation, we rescale interpolation row-wise, either using constant scaling, as in Algorithm 4.6, or the improved iteration scaling, as in Algorithm 4.7.

Finally, we present the two-level AMGr solution phase in Algorithm 4.8. This includes either F- or FCF-relaxation both before and after the coarse-grid correction phase, as well as a standard Galerkin coarse-grid correction. While we present the algorithm without implementation details, we note that the algorithms here can be implemented in either the "natural" ordering of matrix A, or in the "permuted" ordering given in (4.3). In many ways, it is simpler to implement the algorithm after permuting A into its F-C ordering.

Algorithm 4.5 Interpolation Operator

function INTERPOLATION($\hat{A}$, $\hat{A}_{FF}$, $\hat{A}_{FC}$, F, C)
  $\mathcal{Z} \leftarrow \mathcal{S}\left(\hat{A}_{FC} + \hat{A}_{FF}\hat{A}_{FC}\right)$
  $W \leftarrow$ SPAI($\hat{A}_{FF}, \hat{A}_{FC}, \mathcal{Z}$)
  $W \leftarrow W$ after truncating small entries
  if improved iteration scaling to be used **then**
    $W \leftarrow$ IMPROVED-ITERATION-SCALING($\hat{A}, W, F, C$)
  else
    $W \leftarrow$ CONSTANT-SCALING(W, F, C)
  $P \leftarrow \begin{bmatrix} W \\ I \end{bmatrix}$
  return P

Algorithm 4.6 Constant Scaling

function CONSTANT-SCALING(W, F, C)
  $\boldsymbol{s} \leftarrow \mathbf{1}_F/(W\mathbf{1}_C)$ ▷ Componentwise division
  $S \leftarrow$ matrix with $\boldsymbol{s}$ on diagonal and elsewhere 0
  $W \leftarrow SW$
  return W

4.5 Results

While the algorithms given above were derived by focusing on performance for finite-element discretizations of anisotropic diffusion equations on uniform grids, we emphasize in this section that this methodology appears to have much wider applicability. Here, we first evaluate the approach on isotropic diffusion equations, on both structured and unstructured grids. Furthermore, we consider results for a classic "four-quadrant" problem, with piecewise constant diffusion and reaction coefficients on a uniform grid, and for constant-coefficient anisotropic diffusion on an unstructured grid. In all results before Section 4.5.4, we use the simulated annealing coarsening algorithm described above with $\eta = 0.65$.

Algorithm 4.7 Improved Iteration Scaling

function IMPROVED-ITERATION-SCALING($\hat{A}, W, F, C$)
  $n_{\text{wj}} \leftarrow 5$, $\omega \leftarrow 2/3$, $D \leftarrow$ diagonal of $\hat{A}$
  $\boldsymbol{z} \leftarrow \mathbf{1}$
  for $i \leftarrow 1, \ldots, n_{\text{wj}}$ **do**
    $\boldsymbol{z} \leftarrow (I - \omega D^{-1}\hat{A})\boldsymbol{z}$
  $\boldsymbol{s} \leftarrow \boldsymbol{z}_F/(W\boldsymbol{z}_C)$ ▷ Componentwise division
  $S \leftarrow$ matrix with $\boldsymbol{s}$ on diagonal and elsewhere 0
  $W \leftarrow SW$
  return W

4.5.1 Isotropic Poisson problem

In this section, we consider the isotropic diffusion equation $-\Delta u = f$, with Dirichlet boundary conditions, first on uniform meshes of the unit square domain. As a benchmark, the first block column in Table 4.12 presents convergence for the classical AMGr algorithm using a diagonal approximation, D_{FF}, to A_{FF} in both relaxation and interpolation with only F-relaxation. We note that using $\eta = 0.65$ already yields a positive effect on convergence; using $\eta = 0.56$ (as considered in past work) leads to convergence factors around 0.7, instead of 0.37. In either case, while the convergence factors are bounded away from unity independently of grid size, the convergence is suboptimal for AMG on the model Poisson equation on a uniform grid. The remaining columns of Table 4.12 present results for the algorithm of Section 4.4.2, demonstrating substantial improvement in two-grid convergence and reasonable three-level convergence. We also note that the two-level generalized AMGr algorithm using F-relaxation in place of FCF-relaxation also offers reasonable convergence factors of about 0.1. Here, we see that while the grid complexities for these cycles are relatively reasonable, the operator complexities are high, above 3.0 for most of the three-level cycles.

To reduce the computational complexities, we truncate the smaller elements in the interpolation operator as discussed above. A critical question in using interpolation truncation is the choice of the value of parameter ζ. The left plot of Figure 4.2 shows the effects of varying this parameter for the 32×32 uniform grid. For small values of ζ, we observe large complexities, but also excellent two-level convergence factors. As ζ increases past 0.2, so do the convergence factors, yet the complexity continues

Algorithm 4.8 Generalized AMGr Solution Phase

function TWO-LEVEL($A, \boldsymbol{b}, \boldsymbol{u}, F, C, P, A_C, D_{FF}^{\text{inv}}, \sigma_F, D_{CC}^{\text{inv}}, \sigma_C$)
  $\boldsymbol{u} \leftarrow F$-relaxation on $A\boldsymbol{u} = \boldsymbol{b}$
  if C-relaxation to be used **then**
    $\boldsymbol{u} \leftarrow C$-relaxation on $A\boldsymbol{u} = \boldsymbol{b}$
    $\boldsymbol{u} \leftarrow F$-relaxation on $A\boldsymbol{u} = \boldsymbol{b}$
  $\boldsymbol{r}_C \leftarrow P^T(\boldsymbol{b} - A\boldsymbol{u})$
  $\boldsymbol{e}_C \leftarrow$ solution of $A_C\boldsymbol{e}_C = \boldsymbol{r}_C$
  $\boldsymbol{u} \leftarrow \boldsymbol{u} + P\boldsymbol{e}_C$
  $\boldsymbol{u} \leftarrow F$-relaxation on $A\boldsymbol{u} = \boldsymbol{b}$
  if C-relaxation to be used **then**
    $\boldsymbol{u} \leftarrow C$-relaxation on $A\boldsymbol{u} = \boldsymbol{b}$
    $\boldsymbol{u} \leftarrow F$-relaxation on $A\boldsymbol{u} = \boldsymbol{b}$
  return $\boldsymbol{u}$

drop. If we were solely concerned with convergence, we might conclude that this is the optimal value of ζ, since it yields the lowest complexity while retaining the best-possible convergence factor. However, to better balance cost vs. complexity, we prefer to take $\zeta = 0.25$, where we approximately minimize the two-level complexity, while still retaining an acceptable convergence factor. Table 4.13 shows two- and three-grid performance as we vary grid size with $\zeta = 0.25$. We see substantial improvements in complexity, with two-level complexities now similar to those of the classical AMGr algorithm in Table 4.12, and three-level grid complexities now about 2.2, instead of over 3.0. At the same time, excellent two-level convergence factors are maintained, and there is only a slight impact on three-level convergence factors.

An important consideration for algebraic multigrid methods is whether or not they retain their performance as we transition from structured to unstructured grids. Hence, our next problem considers the same isotropic diffusion operator, but discretized using piecewise linear finite elements on unstructured triangulations of the square domain, $[-1, 1]^2$. We construct grids by starting from an unstructured grid, performing several steps of uniform refinement, then smoothing the resulting grids. Here, we consider three levels of refinement, generating meshes with 1433, 5617, and 22 241 DoFs. We again study the effects of varying the truncation parameter, ζ, at right of Figure 4.2, and conclude that taking $\zeta = 0.25$ again gives a good trade-off between convergence and complexity. Table 4.14 shows the resulting two- and three-grid

Table 4.12: AMGr convergence factors and complexities for **isotropic FE discretization on structured grids**. Results for classical (diagonal D_{FF}) AMGr appear in the first block column, followed by those for the two-level and three-level SPAI-based algorithm in Section 4.4.2 in following block columns, using $\zeta = 0$.

	Classical Two-level cycle			Two-level cycle			Three-level cycles			
Grid size	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}	ρ_V	ρ_W	C_{grid}	C_{op}
16×16	0.365	1.36	1.71	0.041	1.36	2.22	0.097	0.044	1.49	2.46
32×32	0.375	1.38	1.80	0.041	1.38	2.64	0.094	0.042	1.53	3.07
64×64	0.373	1.40	1.86	0.039	1.40	2.86	0.102	0.042	1.55	3.37
128×128	0.367	1.41	1.90	0.040	1.41	2.97	0.120	0.042	1.57	3.54

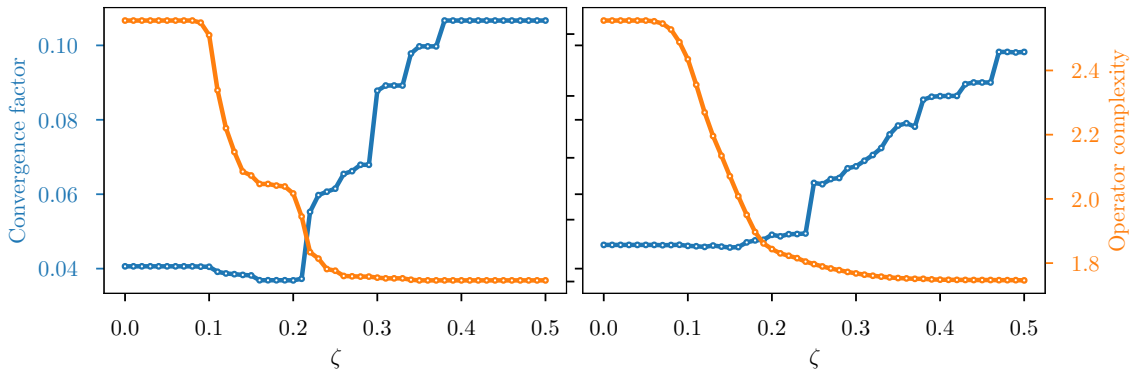


Figure 4.2: Trade-off between two-level convergence factor and operator complexities as a function of ζ, for isotropic Poisson on a uniform 32×32 grid (at left) and on the unstructured triangulation with 1433 DoFs (at right).

Table 4.13: AMGr convergence factors and complexities for isotropic FE discretization on structured grids, using the two-level and three-level SPAI-based algorithm in Section 4.4.2, **with** $\zeta = 0.25$.

	Two-level cycle			Three-level cycles			
Grid size	ρ	C_{grid}	C_{op}	ρ_V	ρ_W	C_{grid}	C_{op}
16×16	0.053	1.36	1.73	0.099	0.058	1.49	1.90
32×32	0.061	1.38	1.84	0.140	0.071	1.52	2.07
64×64	0.065	1.40	1.91	0.151	0.074	1.55	2.20
128×128	0.069	1.41	1.95	0.157	0.079	1.56	2.26

Table 4.14: AMGr convergence factors and complexities for isotropic FE discretization on **unstructured grids**, using the two-level and three-level SPAI-based algorithm in Section 4.4.2. Results in the left-most block column show two-level results with no interpolation truncation ($\zeta = 0$), while the other block columns show results with $\zeta = 0.25$.

	without truncation			with interpolation truncation						
	Two-level cycle			Two-level cycle			Three-level cycles			
#DoF	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}	ρ_V	ρ_W	C_{grid}	C_{op}
1433	0.046	1.36	2.56	0.063	1.36	1.80	0.155	0.064	1.50	2.17
5617	0.135	1.36	2.53	0.132	1.36	1.75	0.167	0.127	1.50	2.17
22241	0.167	1.37	2.49	0.167	1.37	1.78	0.322	0.186	1.52	2.22

convergence factors and operator complexities for the new AMGr algorithm applied to these problems. While the convergence factors are somewhat larger than those for the uniform-grid discretization, they remain acceptable for AMG convergence for an isotropic diffusion operator. Furthermore, we see the efficacy of interpolation truncation in reducing the operator complexity while maintaining acceptable convergence factors.

4.5.2 Four-quadrant problems

Next, we consider a family of two-dimensional anisotropic diffusion problems by adding a reaction term to Equation (4.7), giving

$$-\nabla \cdot \boldsymbol{K}(x,y)\nabla u(x,y) + c(x,y)u(x,y) = b(x,y) \tag{4.10}$$

in the domain $[0,1]\times[0,1]$ with Dirichlet boundary conditions. The tensor coefficient is chosen as $\boldsymbol{K}(x,y) = QHQ^T$, where $Q = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$, and $H = \begin{bmatrix} 1 & 0 \\ 0 & \delta \end{bmatrix}$, where θ specifies the direction of anisotropy in the problem and δ specifies its strength. We note that the convention for H used here is different than that in Equation (4.8), to make the problems consistent with those used in other works. We partition the domain $[0,1]\times[0,1]$ into four equal quadrants and consider constant values of θ, δ and c within each quadrant, but with different values in different quadrants of the domain. The four-quadrant problem is common in AMG literature, and we consider three

different problems within this class, with coefficient values shown below in Figure 4.3. Problem 1 is similar to the problem in Chapter 8 in the book by Briggs, Henson, and McCormick [20], with no reaction term, large contrasts in the anisotropy strength, and non-grid-aligned diffusion in just one quadrant. Problem 2 is the 2D-4Reg problem from Brannick and Falgout [19], with a large reaction coefficient in one quadrant, but a small contrast in anisotropy strength and only grid-aligned anisotropy. Finally, Problem 3 is constructed to provide a more significant challenge, including a large reaction coefficient in one quadrant, a large contrast in anisotropy strength, and anisotropy directions in two quadrants that are neither aligned with the grid nor with the grid diagonal.

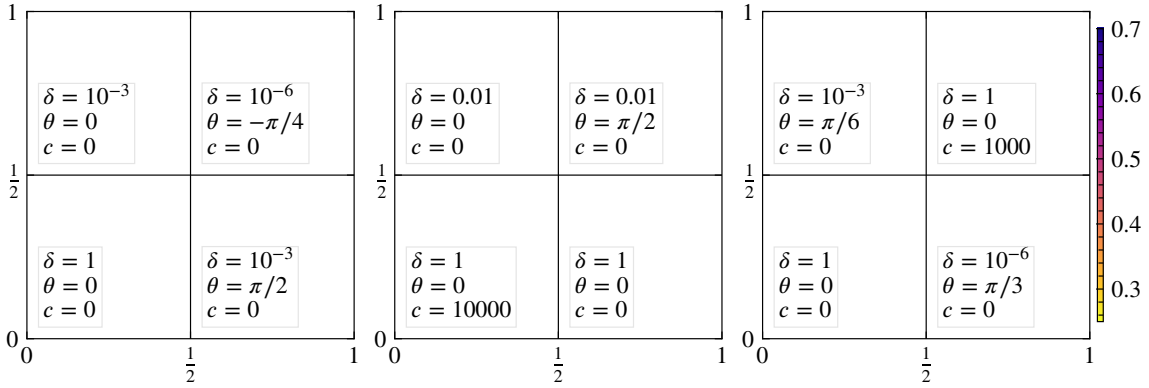


Figure 4.3: Visualization of the strong connections for the four-quadrant problems; Problem 1 (left), Problem 2 (middle), and Problem 3 (right).

Two-level and three-level AMGr performance for these problems is shown in Tables 4.15 and 4.16, respectively. We note that the two-level performance for Problems 1 and 2 is generally good, both in terms of convergence factor and complexity, while Problem 3 is clearly a harder problem. Indeed, both V- and W-cycle convergence continues to perform well for Problem 2 in the three-level results in Table 4.16, with convergence outperforming that reported for the 65×65 grid in Table 4.2 in Brannick and Falgout [19], with comparable grid and operator complexities to the compatible relaxation AMG solver proposed there, and much better complexities than those reported there for BoomerAMG [36]. Problem 3 is clearly more taxing for AMG, yet the proposed generalized AMGr approach offers acceptable convergence in all cases. Whether further improvement to these results is possible (or the performance degradation with grid size can be attenuated using Krylov acceleration) is left for future

Table 4.15: **Two-level** AMGr convergence factors and complexities for the **four-quadrant problem** using the SPAI-based algorithm in Section 4.4.2, with $\zeta = 0.25$.

	Problem 1			Problem 2			Problem 3		
Grid size	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}	ρ	C_{grid}	C_{op}
17×17	0.388	1.35	1.49	0.602	1.26	1.32	0.062	1.25	1.38
33×33	0.433	1.34	1.51	0.581	1.26	1.37	0.266	1.33	1.62
65×65	0.413	1.36	1.56	0.595	1.27	1.41	0.491	1.39	1.78
129×129	0.420	1.36	1.57	0.599	1.36	1.60	0.692	1.39	1.80

Table 4.16: **Three-level** AMGr convergence factors and complexities for the **four-quadrant problem** using the SPAI-based algorithm in Section 4.4.2, with $\zeta = 0.25$.

	Problem 1				Problem 2				Problem 3			
Grid size	ρ_V	ρ_W	C_{grid}	C_{op}	ρ_V	ρ_W	C_{grid}	C_{op}	ρ_V	ρ_W	C_{grid}	C_{op}
17×17	0.404	0.390	1.48	1.65	0.608	0.602	1.34	1.39	0.095	0.064	1.32	1.44
33×33	0.474	0.440	1.46	1.68	0.584	0.581	1.35	1.48	0.314	0.273	1.43	1.78
65×65	0.523	0.422	1.49	1.77	0.598	0.595	1.36	1.54	0.539	0.499	1.52	2.05
129×129	0.599	0.433	1.49	1.80	0.604	0.600	1.46	1.75	0.740	0.702	1.54	2.12

work.

4.5.3 Unstructured anisotropic diffusion

Next, we consider the anisotropic diffusion problem in (4.10) with Dirichlet boundary conditions, $c = 0$, $\theta = \pi/3$, and $\delta = 0.01$, on an unstructured triangulation of the unit square taken from Brannick and Falgout [19], where the problem is labeled as 2D-M2-RLap. As in Table 4.2 from Brannick and Falgout [19], we consider three refinements of the unstructured mesh for this problem, yielding discretized problems with 798, 3109, and 12 273 DoFs, respectively. The mesh containing 798 DoFs (and its coarsening using $\eta = 0.65$) is shown in Figure 4.4. Table 4.17 presents two- and three-level convergence results for the generalized AMGr algorithm applied to this problem. For comparison, we note that Table 4.2 of Brannick and Falgout [19] reports higher convergence factors (up to 0.95 on the finest grid) for CR-AMG applied to this problem, but at lower grid and operator complexities. Compared to the BoomerAMG results presented in the same table, we see comparable convergence (0.57 for the finest

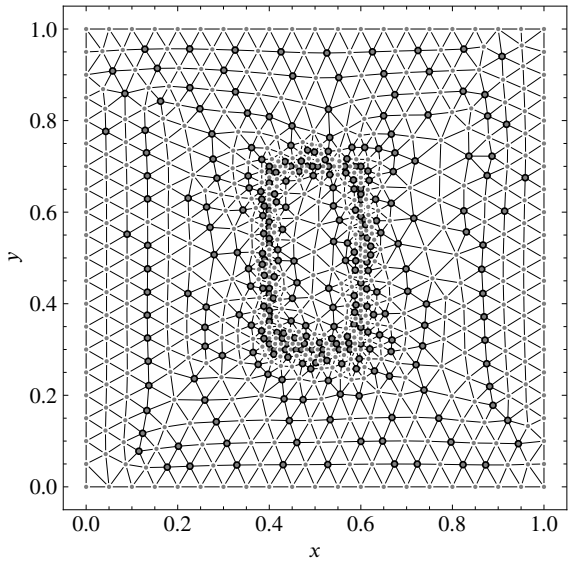


Figure 4.4: Unstructured triangulation containing 798 points from Brannick and Falgout [19], and its partitioning using $\eta = 0.65$. Fine-grid DoFs (points in Ω) are denoted by filled grey dots; those that are in C (282 points) are marked with black circles.

grid) at lower complexity (2.6 at the finest grid, albeit for a multilevel cycle, not a three-level cycle). Compared to classical AMGr applied to this problem, as given in Table 6 of Zaman et al. [60], we see substantial improvement in convergence factors (compared to values of 0.8–0.9 on the finest grid) and lower complexities in these results.

4.5.4 Multilevel Results

Two major obstacles remain in Algorithm 4.4 for transitioning from the two- and three-level cycles studied above to standard multilevel cycles. First of all, we have (until now) focused on the use of simulated annealing for determining the partitioning of A and $\hat{A}$ into the F and C sets. While this is very effective, it is also very costly, as many SA steps are required to generate near-optimal partitionings using this algorithm. Thus, we switch here to using the greedy coarsening algorithm of MacLachlan and Saad [39], which is much more efficient, but generates poorer-quality partitions. To compensate, we investigate the effect of the diagonal dominance parameter, η, on the complexities and convergence of the resulting multilevel hierarchies, in order to

Table 4.17: Two- and three-level AMGr convergence factors and complexities for the **anisotropic diffusion problem on unstructured meshes** using the SPAI-based algorithm in Section 4.4.2, with $\zeta = 0.25$.

	Two-level cycle			Three-level cycles			
#DoF	ρ	C_{grid}	C_{op}	ρ_V	ρ_W	C_{grid}	C_{op}
798	0.516	1.35	1.64	0.586	0.534	1.49	1.95
3109	0.607	1.37	1.64	0.676	0.621	1.51	1.99
12273	0.601	1.37	1.65	0.707	0.639	1.51	2.01

attenuate some of the complexity growth that we observe in the initial results.

The second major obstacle is the calculation of extremal eigenvalues in Lines 8 and 11 of Algorithm 4.4, which has additional heavy computational cost. To eliminate this, we replace the optimal calculation of σ_F and σ_C with a common heuristic estimate of the optimal regularization parameter. Knowing that A_{FF} and A_{CC} are both positive-definite matrices, we expect that D_{FF}^{inv} and D_{CC}^{inv} are as well. If this is the case, the spectra of $D_{FF}^{\text{inv}}A_{FF}$ and $D_{CC}^{\text{inv}}A_{CC}$ are guaranteed to be contained in the intervals from 0 to their largest eigenvalues, which can be estimated by their maximum absolute row sums (using Geršgorin's theorem). While the F-relaxation originally used in AMGr targets an optimal reduction over all modes by estimating both ends of the spectrum of $D_{FF}^{\text{inv}}A_{FF}$, we propose a simpler heuristic of choosing σ_F and σ_C to be 3/2 divided by the maximum absolute row sum of $D_{FF}^{\text{inv}}A_{FF}$ and $D_{CC}^{\text{inv}}A_{CC}$, respectively. The choice of weight 3/2 in this heuristic reflects the expectation that these matrices are well-conditioned, so we need not use a weight as large as 2 (which would be optimal if we estimate the smallest eigenvalues as 0), but that they are far from perfectly conditioned, so the weight should be larger than 1. Numerical tests confirm that using weight 3/2 is a good compromise — in some cases, some improvements are possible with larger weights, but this leads to greatly degraded performance in some cases as well.

As a comparison with the final three-level results in Table 4.11. Table 4.18 shows convergence factors, complexities, and number of levels in the multigrid hierarchies (n_l) for the multilevel algorithm. These results show notable degradation in both operator and grid complexities, due to the use of greedy coarsening with $\eta = 0.65$ in contrast with the simulated annealing coarsening used in the previous results. Nonetheless, we observe excellent W-cycle convergence factors in all cases (outper-

Table 4.18: **Multi-level** AMGr convergence factors and corresponding complexities for anisotropic FE discretization using **greedy coarsening** with $\eta = 0.65$. F-nodes are interpolated using the SPAI approximation to $\hat{A}_{FF}^{-1}\hat{A}_{FC}$, postprocessed to exactly interpolate a relaxed vector, and the SPAI approximations to $\hat{A}_{FF}^{-1}$ and $\hat{A}_{CC}^{-1}$ are used for FCF-relaxation, where $\hat{A}$ is the "lumped" matrix of strong connections computed from A. Interpolation truncation with $\zeta = 0.2$ is employed. **Estimates of the eigenvalues are used in relaxation.**

	$\theta = 0$					$\theta = \pi/6$					$\theta = \pi/4$				
Grid size	ρ_V	ρ_W	C_{grid}	C_{op}	n_{l}	ρ_V	ρ_W	C_{grid}	C_{op}	n_{l}	ρ_V	ρ_W	C_{grid}	C_{op}	n_{l}
32×32	0.205	0.186	1.78	1.73	4	0.312	0.136	1.76	2.24	4	0.383	0.170	1.74	2.11	4
64×64	0.230	0.187	1.84	1.81	6	0.515	0.139	1.88	2.50	6	0.460	0.149	1.87	2.45	6
128×128	0.232	0.188	1.91	1.88	7	0.641	0.163	1.93	2.64	8	0.668	0.285	1.92	2.61	8
256×256	0.242	0.186	1.95	1.93	8	0.722	0.180	1.96	2.72	10	0.740	0.346	1.96	2.72	10

forming the earlier results for $\theta = 0$ and $\theta = \pi/6$), and consistent V-cycle convergence factors. In experiments not reported here, we compared convergence to the case of using exact eigenvalues and found little difference in convergence overall. Notably, when using the multigrid cycles as preconditioners for conjugate gradient, using the heuristic choice incurs at most 3 additional iterations over using cycles based on the exact eigenvalue computation.

Similar results are shown in Table 4.19 for the four-quadrant problems from Section 4.5.2, for comparison with Table 4.16. Again, we note that the complexities are much higher than those reported earlier using the simulated annealing coarsening algorithm, but that this added complexity pays off in improved multilevel convergence. For Problem 2, we again compare to the results presented in Table 4.2 by Brannick and Falgout [19], and see that this coarsening achieves comparable complexities to those reported there for 33×33 and 65×65 grids, but much better convergence. Overall, we again see grid-independent W-cycle convergence for each problem, but growth in V-cycle convergence factors. When run as preconditioners for CG, we find that W-cycles lead to convergence in 5–10 iterations (more for Problem 1, fewer for Problems 2 and 3), and V-cycle convergence in up to 14 iterations (again, with Problem 1 requiring most, and Problem 3 requiring fewest).

Finally, we present results for the anisotropic diffusion problem on unstructured meshes considered in Section 4.5.3 and Table 4.17. Here, to explore the connection

Table 4.19: **Multi-level** AMGr convergence factors and complexities for the **four-quadrant problem** using the SPAI-based algorithm in Section 4.4.2, with $\zeta = 0.25$. **Greedy coarsening and the heuristic eigenvalue estimates are used.**

	Problem 1					Problem 2					Problem 3				
Grid size	ρ_V	ρ_W	C_{grid}	C_{op}	n_l	ρ_V	ρ_W	C_{grid}	C_{op}	n_l	ρ_V	ρ_W	C_{grid}	C_{op}	n_l
33×33	0.541	0.412	1.77	2.08	4	0.429	0.334	1.55	1.73	4	0.202	0.097	1.66	2.17	4
65×65	0.602	0.419	1.87	2.30	6	0.435	0.263	1.65	1.92	6	0.414	0.128	1.84	2.59	6
129×129	0.698	0.421	1.93	2.43	8	0.323	0.256	1.83	2.34	8	0.621	0.151	1.92	2.82	8
257×257	0.783	0.420	1.97	2.49	11	0.367	0.258	1.95	2.63	11	0.769	0.210	1.96	2.95	10

Table 4.20: **Multi-level** AMGr convergence factors and complexities for **anisotropic diffusion problem on unstructured meshes** using the SPAI-based algorithm in Section 4.4.2, with $\zeta = 0.25$. **Greedy coarsening, for three values of η, and the heuristic eigenvalue estimates are used.**

	$\eta = 0.56$					$\eta = 0.60$					$\eta = 0.65$				
#DoF	ρ_V	ρ_W	C_{grid}	C_{op}	n_l	ρ_V	ρ_W	C_{grid}	C_{op}	n_l	ρ_V	ρ_W	C_{grid}	C_{op}	n_l
798	0.612	0.560	1.54	2.06	4	0.601	0.529	1.66	2.37	4	0.537	0.467	1.76	2.56	4
3109	0.732	0.665	1.66	2.36	5	0.712	0.591	1.77	2.66	6	0.686	0.577	1.90	2.98	6
12273	0.777	0.653	1.71	2.51	7	0.762	0.630	1.83	2.87	8	0.742	0.576	1.98	3.33	9

between the diagonal dominance parameter, η, and the resulting complexities and convergence factors, we consider $\eta = 0.65$ as before, along with $\eta = 0.60$ and $\eta = 0.56$. Table 4.20 shows that, as expected, complexities decrease and convergence factors generally increase as η gets smaller, but that significant improvements in complexity are possible by using smaller η without sacrificing substantial convergence. In particular, comparing results for $\eta = 0.56$, we observe modest increases in grid complexity in comparison with those in Table 4.17, possibly attributed to the increase from three-level to multi-level cycles. Comparing these results to that presented in Table 4.2 by Brannick and Falgout [19], we observe complexities better than those reported for BoomerAMG for these problems, albeit with slightly worse convergence, and slightly worse than those reported for compatible relaxation, but with better convergence.

4.6 Conclusions and future work

Reduction-based AMG methods have been proposed and studied in many settings over the past 15 years, building effective solvers that can be more closely related to AMG convergence theory than many other heuristic methods. In this paper, we aim to improve the practical performance of AMGr approaches by targeting tools that can greatly improve performance for anisotropic diffusion equations. Through extensive numerical results, we show that the combination of using SPAI [34, 35] to approximate A_{FF}^{-1} along with tools to control sparsity leads to effective solvers for both anisotropic and isotropic diffusion operators on structured and unstructured grids. In our view, this work points to weaknesses in the existing theory for AMGr-type methods, where approximations to $A_{FF}^{-1}A_{FC}$ have not been considered (to our knowledge) in any context, providing an opportunity for future work. This may also provide new insights into desirable properties of SPAI-like approximations in this context. Additionally, further experiments are needed to see how to adapt the methodology proposed here to an even broader set of challenging problems, including the indefinite Helmholtz equation and convection-dominated flows.

Acknowledgments

The work of S.P.M. was partially supported by an NSERC Discovery Grant. This work does not have any conflicts of interest.

Bibliography

[1] Allison H. Baker, Robert D. Falgout, Tzanio V. Kolev, and Ulrike Meier Yang. Multigrid smoothers for ultraparallel computing. *SIAM Journal on Scientific Computing*, 33(5):2864–2887, 2011. doi: 10.1137/100798806.

[2] Nathan Bell, Luke N. Olson, and Jacob Schroder. PyAMG: Algebraic multigrid solvers in python. *Journal of Open Source Software*, 7(72):4142, 2022. doi: 10.21105/joss.04142. URL `https://doi.org/10.21105/joss.04142`.

[3] Maurice W Benson. Iterative solution of large scale linear systems. M.sc. thesis, Lakehead University, Thunder Bay, Canada, 1973.

[4] Maurice W. Benson. Frequency domain behavior of a set of parallel multigrid smoothing operators. *International Journal of Computer Mathematics*, 36(1-2): 77–88, 1990. doi: 10.1080/00207169008803912.

[5] Maurice W. Benson and Paul O. Frederickson. Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems. *Utilitas Math.*, 22:127–140, 1982. ISSN 0315-3681.

[6] Michele Benzi and Miroslav Tuma. A comparative study of sparse approximate inverse preconditioners. *Applied Numerical Mathematics*, 30(2-3):305–340, 1999.

[7] Michele Benzi, Carl D Meyer, and Miroslav Tůma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM Journal on Scientific Computing*, 17(5):1135–1149, 1996.

[8] Michele Benzi, Jane K Cullum, and Miroslav Tuma. Robust approximate inverse preconditioning for the conjugate gradient method. *SIAM Journal on Scientific Computing*, 22(4):1318–1332, 2000.

[9] Matthias Bollhöfer. Adapted sparse approximate inverse smoothers in algebraic multilevel methods. Technical Report Preprint 759-2002, Institute of Mathematics, Technische Universität Berlin, 2002.

[10] Matthias Bollhöfer and Volker Mehrmann. Algebraic multilevel methods and sparse approximate inverses. *SIAM Journal on Matrix Analysis and Applications*, 24(1):191–218, 2002. ISSN 0895-4798. doi: 10.1137/S0895479899364441.

[11] Matthias Bolten, Thomas K Huckle, and Christos D Kravvaritis. Sparse matrix approximations for multigrid methods. *Linear Algebra and its Applications*, 502: 58–76, 2016.

[12] A. Brandt, S. F. McCormick, and J. W. Ruge. Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations. Institute for Computational Studies, Colorado State University, 1982.

[13] A. Brandt, S. F. McCormick, and J. W. Ruge. Algebraic multigrid (AMG) for sparse matrix equations. In D. J. Evans, editor, *Sparsity and Its Applications*, pages 257–284. Cambridge University Press, Cambridge, 1984.

[14] Achi Brandt, James Brannick, Karsten Kahl, and Ira Livshits. An algebraic distances measure of amg strength of connection. *arXiv preprint arXiv:1106.5990*, 2011.

[15] J. Brannick, Y. Chen, J. Kraus, and L. Zikatanov. Algebraic multilevel preconditioners for the graph Laplacian based on matching in graphs. *SIAM Journal on Numerical Analysis*, 51(3):1805–1827, 2013. ISSN 0036-1429. doi: 10.1137/120876083. URL `https://doi.org/10.1137/120876083`.

[16] James Brannick, Marian Brezina, S MacLachlan, T Manteuffel, S McCormick, and J Ruge. An energy-based amg coarsening strategy. *Numerical Linear Algebra with Applications*, 13(2-3):133–148, 2006.

[17] James Brannick, A Frommer, Karsten Kahl, Scott MacLachlan, and Ludmil Zikatanov. Adaptive reduction-based multigrid for nearly singular and highly disordered physical systems. *Electronic Transactions on Numerical Analysis*, 37: 276–295, 2010.

[18] James Brannick, Yao Chen, Johannes Kraus, and Ludmil Zikatanov. An algebraic multigrid method based on matching in graphs. In *Domain Decomposition Methods in Science and Engineering XX*, volume 91 of *Lecture Notes in Computational Science and Engineering*, pages 143–150. Springer, Heidelberg, 2013. doi: 10.1007/978-3-642-35275-1_15. URL `https://doi.org/10.1007/978-3-642-35275-1_15`.

[19] James J. Brannick and Robert D. Falgout. Compatible relaxation and coarsening in algebraic multigrid. *SIAM Journal on Scientific Computing*, 32(3):1393–1416, 2010.

[20] William L Briggs, Van Emden Henson, and Steve F McCormick. *A multigrid tutorial*. SIAM, 2000.

[21] Oliver Bröker. *Parallel multigrid methods using sparse approximate inverses*. Ph.d. thesis, ETH Zurich, Zurich, Switzerland, 2003.

[22] Oliver Bröker and Marcus J Grote. Sparse approximate inverse smoothers for geometric and algebraic multigrid. *Applied Numerical Mathematics*, 41(1):61–80, 2002.

[23] Oliver Bröker, Marcus J. Grote, Carsten Mayer, and Arnold Reusken. Robust parallel smoothing for multigrid via sparse approximate inverses. *SIAM Journal on Scientific Computing*, 23(4):1396–1417, 2001. ISSN 1064-8275. doi: 10.1137/S1064827500380623.

[24] Edmond Chow and Yousef Saad. Approximate inverse preconditioners via sparse-sparse iterations. *SIAM Journal on Scientific Computing*, 19(3):995–1023, 1998.

[25] Hans De Sterck, Ulrike Meier Yang, and Jeffrey J. Heys. Reducing complexity in parallel algebraic multigrid preconditioners. *SIAM Journal on Matrix Analysis and Applications*, 27(4):1019–1039, 2006. ISSN 0895-4798.

[26] Hans De Sterck, Robert D. Falgout, Joshua W. Nolting, and Ulrike Meier Yang. Distance-two interpolation for parallel algebraic multigrid. *Numerical Linear Algebra with Applications*, 15(2-3):115–139, 2008. ISSN 1070-5325.

[27] R.D. Falgout and P.S. Vassilevski. On generalizing the AMG framework. *SIAM Journal on Numerical Analysis*, 42(4):1669–1693, 2004.

[28] R.D. Falgout and U.M. Yang. Hypre: A library of high performance preconditioners. In P.M.A. Sloot, C.J.K. Tan, J.J. Dongarra, and A.G. Hoekstra, editors, *Computational Science - ICCS 2002: International Conference, Amsterdam, The Netherlands, April 21-24, 2002. Proceedings, Part III*, number 2331 in Lecture Notes in Computer Science, pages 632–641. Springer-Verlag, 2002.

[29] Robert D. Falgout, Stephanie Friedhoff, Tzanio V. Kolev, Scott P. MacLachlan, and Jacob B. Schroder. Parallel time integration with multigrid. *SIAM Journal on Scientific Computing*, 14(1):951–952, 2014.

[30] Christos K Filelis-Papadopoulos and George A Gravvanis. Parallel multigrid algorithms based on generic approximate sparse inverses: an smp approach. *The Journal of Supercomputing*, 67(2):384–407, 2014.

[31] Paul O Frederickson. *Fast approximate inversion of large sparse linear systems.* Lakehead University, Department of Mathematical Sciences, 1975.

[32] Florian Gossler and Reinhard Nabben. On amg methods with f-smoothing based on chebyshev polynomials and their relation to amgr. *Electronic Transactions on Numerical Analysis*, 45:146–159, 2016.

[33] George A Gravvanis, Christos K Filelis-Papadopoulos, and Paschalis I Matskanidis. Algebraic multigrid methods based on generic approximate inverse matrix techniques. *Computer Modeling in Engineering & Sciences*, 100(4):323–345, 2014.

[34] Marcus J Grote and Thomas Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM Journal on Scientific Computing*, 18(3):838–853, 1997.

[35] Stuart C. Hawkins and Ke Chen. An implicit wavelet sparse approximate inverse preconditioner. *SIAM Journal on Scientific Computing*, 27(2):667–686, 2005. doi: 10.1137/S1064827503423500.

[36] V.E. Henson and U.M. Yang. BoomerAMG: A parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, 41:155–177, 2002.

[37] L Yu Kolotilina and A Yu Yeremin. Factorized sparse approximate inverse preconditionings i. theory. *SIAM Journal on Matrix Analysis and Applications*, 14 (1):45–58, 1993.

[38] S. MacLachlan and Y. Saad. Greedy coarsening strategies for nonsymmetric problems. *SIAM Journal on Scientific Computing*, 29(5):2115–2143, 2007.

[39] S. MacLachlan and Y. Saad. A greedy strategy for coarse-grid selection. *SIAM Journal on Scientific Computing*, 29(5):1825–1853, 2007. doi: 10.1137/060654062.

[40] S. MacLachlan, T. Manteuffel, and S. McCormick. Adaptive reduction-based AMG. *Numerical Linear Algebra with Applications*, 13:599–620, 2006.

[41] Thomas A Manteuffel, John Ruge, and Ben S Southworth. Nonsymmetric algebraic multigrid based on local approximate ideal restriction (ℓAIR). *SIAM Journal on Scientific Computing*, 40(6):A4105–A4130, 2018.

[42] Thomas A Manteuffel, Steffen Münzenmaier, John Ruge, and Ben Southworth. Nonsymmetric reduction-based algebraic multigrid. *SIAM Journal on Scientific Computing*, 41(5):S242–S268, 2019.

[43] Christian Mense and Reinhard Nabben. On algebraic multilevel methods for nonsymmetric systems-convergence results. *Electronic Transactions on Numerical Analysis*, 30:323–345, 2008.

[44] Gérard Meurant. Numerical experiments with algebraic multilevel preconditioners. *Electronic Transactions on Numerical Analysis*, 12:1–65, 2001.

[45] Gérard Meurant. A multilevel ainv preconditioner. *Numerical Algorithms*, 29 (1):107–129, 2002.

[46] Arne Nägel, Robert D Falgout, and Gabriel Wittum. Filtering algebraic multigrid and adaptive strategies. *Computing and Visualization in Science*, 11(3):159–167, 2008.

[47] Artem Napov and Yvan Notay. An algebraic multigrid method with guaranteed convergence rate. *SIAM Journal on Scientific Computing*, 34(2): A1079–A1109, 2012. ISSN 1064-8275. doi: 10.1137/100818509. URL `https://doi.org/10.1137/100818509`.

[48] Artem Napov and Yvan Notay. An efficient multigrid method for graph Laplacian systems. *Electronic Transactions on Numerical Analysis*, 45:201–218, 2016.

[49] Artem Napov and Yvan Notay. An efficient multigrid method for graph Laplacian systems II: Robust aggregation. *SIAM Journal on Scientific Computing*, 39(5):S379–S403, 2017. ISSN 1064-8275. doi: 10.1137/16M1071420. URL `https://doi.org/10.1137/16M1071420`.

[50] Luke N Olson, Jacob Schroder, and Raymond S Tuminaro. A new perspective on strength measures in algebraic multigrid. *Numerical Linear Algebra with Applications*, 17(4):713–733, 2010.

[51] M. Ries, U. Trottenberg, and G. Winter. A note on MGR methods. *Linear Algebra and its Applications*, 49:1–26, 1983.

[52] J. W. Ruge and K. Stüben. Algebraic multigrid (AMG). In S. F. McCormick, editor, *Multigrid Methods*, volume 3 of *Frontiers in Applied Mathematics*, pages 73–130. SIAM, Philadelphia, PA, 1987.

[53] K. Stüben. An introduction to algebraic multigrid. In U. Trottenberg, C. Oosterlee, and A. Schüller, editors, *Multigrid*, pages 413–528. Academic Press, London, 2001.

[54] Klaus Stüben. Algebraic multigrid (AMG): Experiences and comparisons. *Applied Mathematics and Computation*, 13(3-4):419–451, 1983. ISSN 0096-3003. doi: 10.1016/0096-3003(83)90023-1.

[55] Paul N. Swarztrauber. The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle. *SIAM Review*, 19(3):490–501, 1977. ISSN 0036-1445. doi: 10.1137/1019071.

[56] Ali Taghibakhshi, Scott MacLachlan, Luke Olson, and Matthew West. Optimization-based algebraic multigrid coarsening using reinforcement learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 12129–12140. Curran Associates, Inc., 2021.

[57] R. S. Varga. *Matrix Iterative Analysis*. Springer Series in Computational Mathematics. Springer, Berlin, 2000. Second Edition.

[58] Christian Wagner. On the algebraic construction of multilevel transfer operators. *Computing*, 65(1):73–95, 2000.

[59] Shun Wang and Eric de Sturler. Multilevel sparse approximate inverse preconditioners for adaptive mesh refinement. *Linear algebra and its applications*, 431 (3-4):409–426, 2009.

[60] Tareq Zaman, Scott P MacLachlan, Luke N Olson, and Matthew West. Coarse-grid selection using simulated annealing. *arXiv preprint arXiv:2105.13280*, 2021.

Chapter 5

Generalizing Lloyd's Algorithm for Graph Clustering

Abstract

[1] Clustering is a commonplace problem in many areas of data science, with applications in biology and bioinformatics, understanding chemical structure, image segmentation, building recommender systems, and many more fields. While there are many different clustering variants (based on given distance or graph structure, probability distributions, or data density), we consider here the problem of clustering nodes in a graph, motivated by the problem of aggregating discrete degrees of freedom in multigrid and domain decomposition methods for solving sparse linear systems. Specifically, we consider the challenge of forming *balanced* clusters in the graph of a sparse matrix for use in algebraic multigrid, although the algorithm has general applicability. Based on an extension of the Bellman-Ford algorithm, we generalize Lloyd's algorithm for partitioning subsets of $\mathbb{R}^n$ to balance the number of nodes in each cluster; this is accompanied by a rebalancing algorithm that reduces the overall *energy* in the system. The algorithm provides control over the number of clusters and leads to "well centered" partitions of the graph. Theoretical results are provided to establish linear complexity and numerical results in the context of algebraic multigrid highlight the benefits of improved clustering.

[1]This work is submitted as "Generalizing Lloyd's Algorithm for Graph Clustering" by Tareq Zaman, Nicolas Nytko, Ali Taghibakhshi, Scott MacLachlan, Luke Olson, and Matthew West.

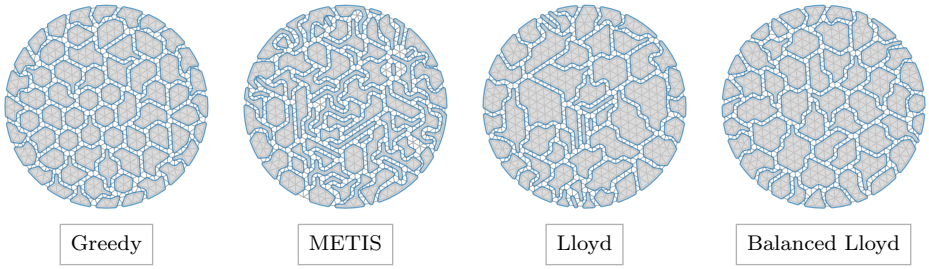


Figure 5.1: Example clusterings.

Keyword: Clustering, Aggregation, Multigrid, Graph Partitioning.

5.1 Introduction

Consider a directed graph $G(V, E, W)$ where V is a set of nodes (or vertices), $V = \{1, \ldots, N_{\text{node}}\}$, and where E is a list of edges given by $E = \{(i, j) \mid W_{i,j} \neq 0\}$ for some weight matrix W. The (sparse) weight matrix W is assumed to have non-negative off-diagonal entries and zero diagonal entries. The goal of this work is to define a set of clusters that minimizes a given energy functional with linear complexity in the number of nodes. In Section 5.3.1, the maximum cluster diameter is used to define the energy.

There are an array of challenges in clustering; the focus here is twofold: (1) developing efficient algorithms where the number of clusters N_{cluster} can be specified; and (2) generating clusterings that are considered "well balanced". As a motivating example, we consider a graph generated from a finite-element discretization on a unit disk with 528 vertices[2]. Figure 5.1 illustrates the clustering of nodes using four different methods that underscore these two challenges. Nearest-neighbor (or *Greedy*) clustering [25] yields 63 clusters in this case. While this simple algorithm lacks control of the number of clusters, the clustering offers a clear balance in the number of nodes per cluster and total diameter of each cluster. In contrast, with $N_{\text{cluster}} = 52$ the spectral-based partitioner METIS [10] yields long clusters (and large diameters).

In this work, we focus on shortest-path based clustering algorithms. *Lloyd* clustering (also known as Lloyd aggregation) [4], for example, uses Bellman-Ford [8] to construct $N_{\text{cluster}} = 52$ clusters based on an initial seeding. Overall clustering qual-

[2]This example is studied in detail in Section 5.4

ity depends *highly* on the initial seeding; often even costly $\mathcal{O}(N_{\text{node}}^3)$ algorithms for seeding, such as *k-means++* [1], do not dramatically improve the final clustering in this case. Lastly, we highlight clustering based on an algorithm introduced here: a balanced form of Lloyd clustering with *rebalancing* to minimize diameter. While standard Lloyd clustering results in both large and small clusters, the cluster shapes with *balanced* Lloyd clustering and *rebalancing* are more consistent.

While there is a long history of aggregation-based multigrid methods (cf. [15, 17, 25, 26]), surprisingly little attention has been paid to the influence of cluster quality on the performance of the resulting algorithm. The *greedy* clustering algorithm originally proposed in [15] has become a standard approach that is used in many codes. Some variants on this approach have been introduced for massively parallel settings; most notably, approaches based on distance-two maximal independent sets in the graph [2, 11, 24]. Both of these approaches make minimal use of the weight matrix, W, aside from using its nonzero pattern to infer binary connectivity data in the graph. In contrast, in [4], Lloyd's algorithm [12] was extended from computing Voronoi diagrams in $\mathbb{R}^n$ to computing clusters in graphs, using the values in W to define graph distances. It is this approach that we extend here.

In this paper, we introduce a general clustering method for use in graph partitioning and algebraic multigrid that provides control of the number of clusters, yields "centered" clusters, and can be implemented with off-the-shelf codes for Bellman-Ford and Floyd-Warshall algorithms [8]. All algorithms are implemented in and are available through the open source package PyAMG [3]. In Section 5.2, we review aggregation-based algebraic multigrid (AMG) and survey the greedy, maximal independent set, and Lloyd clustering algorithms. Section 5.3 introduces *balanced* Lloyd clustering and a *rebalance* algorithm, along with theoretical evidence of convergence and complexity. Finally, Section 5.4 provides numerical evidence in support, expanding the example in Figure 5.1 and others.

Note: throughout the paper and embedded in the algorithms, we make use of the notation listed in Table 5.2.

5.2 Clustering in algebraic multigrid

Algebraic multigrid methods are a family of iterative methods for the solution of sparse linear systems of the form $Au = f$, where A is an $N_{\text{node}} \times N_{\text{node}}$ matrix and u and f are vectors of dimension N_{node}. Like all multigrid methods, they achieve

their efficiency through the use of two complementary processes, known as relaxation and coarse-grid correction. For algebraic multigrid methods, we typically consider a fixed relaxation scheme (such as a stationary weighted Jacobi or Gauss-Seidel iteration on the linear system) and seek to compute a coarse-grid correction process that adequately complements relaxation to lead to an efficient solution algorithm. In aggregation-based methods, the coarse-grid correction process takes the form of first computing a clustering of the fine-grid degrees of freedom (nodes in the graph of the sparse matrix, A), and then computing an interpolation operator from the clustered degrees of freedom to those on the fine grid. Rootnode-based aggregation methods additionally make use of a center that is identified for each cluster [14]. For a more thorough review of algebraic multigrid methods, see Appendix A or [7, 23].

Figure 5.1 illustrates the wide range of clusters that can arise for a single problem. We next detail three common approaches to clustering (used in the context of AMG), before introducing a balanced method in the next section. First, however, we define a clustering or aggregation of $G(V, E, W)$, as in Definition 5.1. We note that the clustering is a non-overlapping covering.

Definition 5.1. *A* clustering *or* aggregation *of the connected graph $G(V, E, W)$ is a pair (m, c), where m_i is the cluster membership of vertex i and c_a is the global index of the center for cluster a. Then m and c have the following properties:*

1. *For each $i \in \{1, \ldots, N_{node}\}$, there exists a unique a with $1 \leq a \leq N_{cluster}$ such that $m_i = a$;*

2. *For each $a \in \{1, \ldots, N_{cluster}\}$, for every (i, j) with $m_i = m_j = a$, there exists a sequence $k_1, \ldots, k_p$ where $m_k = a$ for $k \in \{k_1, \ldots, k_p\}$ and with $(i, k_1), (k_q, k_{q+1}), (k_p, j) \in E$ for $q \in \{1, \ldots, p-1\}$; and*

3. *For each $a \in \{1, \ldots, N_{cluster}\}$, we have $1 \leq c_a \leq N_{node}$ and $m_{c_a} = a$.*

The first point ensures that the clustering is a non-overlapping covering, the second requires that the subgraph over the cluster remains connected, and the third confirms that an element of each cluster is identified as the center for that cluster.

5.2.1 Greedy clustering

Greedy clustering (also known as "greedy aggregation" or "standard aggregation") was first introduced by Míka and Vaněk [15]; we use a close variant. Greedy clustering

consists of two passes over the set of nodes of the graph. In the first pass, for each node, if all neighbors in the graph remain unclustered, then the node becomes a center, forming a cluster from the node and its neighborhood. In the second pass, each unclustered node is included in a neighboring cluster, if possible. If a neighboring cluster is not found, then the unclustered node is considered a center node and the node with its unclustered neighbors form a new cluster. In the case of multiple neighboring clusters, there are several options: arbitrary selection, index, size, or magnitude of the weight can each be used to determine cluster membership. The full greedy algorithm is given in Algorithm 5.1.

Algorithm 5.1 Greedy clustering. See Table 5.2 for variable definitions.

function GREEDY-CLUSTERING(W)
  $m_i \leftarrow 0$ for all $i = 1, \ldots, N_{\text{node}}$ ▷ initially all nodes are unclustered
  $a \leftarrow 0$ ▷ first cluster index
  for $i \leftarrow 1, \ldots, N_{\text{node}}$ **do** ▷ **first pass**
    if $m_i = 0$ and $m_j = 0$ for all j s.t. $W_{i,j} \neq 0$ **then** ▷ unclustered
      $m_i \leftarrow a$ ▷ add i and neighbors to cluster a
      $m_j \leftarrow a$, for all j s.t. $W_{i,j} \neq 0$
      $c_a \leftarrow i$ ▷ mark cluster center
      $a \leftarrow a + 1$ ▷ increment cluster index
  for $i \leftarrow 1, \ldots, N_{\text{node}}$ **do** ▷ **second pass**
    if $m_i = 0$ **then** ▷ unclustered
      if $\exists\ j$ s.t. $W_{i,j} \neq 0$ and $m_j > 0$ **then** ▷ clustered neighbor
        $j \leftarrow \underset{j\,:\,m_j>0}{\operatorname{argmax}} W_{i,j}$ ▷ neighbor with largest weight
        $m_i \leftarrow m_j$
      else ▷ form new cluster
        $m_i \leftarrow a$
        for j such that $W_{i,j} \neq 0$ and $m_j = 0$ **do**
          $m_j \leftarrow a$
        $a \leftarrow a + 1$ ▷ increment cluster index
  return m, c

5.2.2 Maximal independent set based clustering

The greedy algorithm is inherently serial, yet there are two immediate observations. First, any two center nodes of two (distinct) clusters must be more than two edges apart. Second, if an unclustered node is more than two edges from any existing center, then the node is eligible to be a center of a new cluster. Hence, the center nodes from the greedy algorithm represent a distance-2 maximal independent set or MIS(2). This leads to the MIS(2) clustering algorithm, where an MIS(2) over the nodes is first constructed, followed by construction of the clustering using the MIS(2) center nodes. This has been shown to exhibit a high degree of parallelism [2]; see [2, Algorithm 5] for details.

Given a distance-2 maximal independent set, the clustering process is straightforward. In the first step, the index of the cluster representing the center is propagated to its neighbors. This continues in the second step, where the index of the cluster is propagated to the second layer of neighbors; if there are multiple clusters adjacent to an unclustered node, the choice is made arbitrarily (or by index). The algorithm is shown in Algorithm 5.2.

Algorithm 5.2 MIS(2) clustering. See Table 5.2 for variable definitions.

```
1:  function MIS(2)-CLUSTERING(W)
2:      c ← MIS(W, 2)                          ▷ distance-2 independent set
3:      m_i ← 0 for i = 1, ..., N_node
4:      N_cluster ← |c|
5:      for a = 1, ..., N_cluster do           ▷ pass 1: distance-1
6:          i ← c_a                            ▷ index of center for cluster a
7:          m_i ← a                            ▷ set cluster number for center
8:          for j s.t. W_{i,j} ≠ 0 do
9:              m_j ← a                        ▷ set cluster number for neighbors
10:     for i s.t. m_i > 0 do                  ▷ pass 2: distance-2
11:         for j s.t. W_{i,j} ≠ 0 and m_j = 0 do
12:             m_j ← m_i                      ▷ set cluster number for neighbors
13:     return m, c
```

With an appropriate ordering, the first pass of MIS-based and greedy clustering can yield identical clusters. With only minor differences in the second pass, we expect

the clustering patterns to be similar. Indeed, the convergence factors of AMG based on these two clustering strategies are shown to be close in practice [2, Appendix].

5.2.3 Standard Lloyd clustering

A shortcoming of the previous two clustering strategies is the inability to control the coarsening rate: the number of clusters is an outcome of the algorithm, rather than an input. In contrast, Lloyd clustering, introduced in [4], is based on an initial seeding of centers (of any length). Lloyd clustering can be viewed as an extension of Lloyd's algorithm [12] applied to graphs, where an initial random seeding of centers yields Voronoi cells (or a set of nodes closest to each center), followed by a recentering of center locations.

A full algorithm is given in Algorithm 5.3, where a subset of N_{cluster} nodes are randomly selected as the initial centers, input as c. A standard Bellman-Ford algorithm (see Algorithm 5.4 and [8, Section 8.7]) is used to find the distance and index of the closest center; the set of points closest to each center form the initial clustering. Next, the border nodes of each cluster are selected and a modified form of the Bellman-Ford algorithm then identifies the (new) center — see Algorithm 5.5 — by selecting the node of maximum distance to the cluster boundary (with ties selected arbitrarily). The steps are repeated until the algorithm has converged or a maximum number of iterations (given as $T_{\max}$) is reached.

Algorithm 5.3 Lloyd clustering algorithm. See Table 5.2 for variable definitions.

```
1: function LLOYD-CLUSTERING(W, c, T_max)
2:     t = 0
3:     repeat
4:         m, d ← BELLMAN-FORD(W, c)                ▷ find closest centers
5:         c ← MOST-INTERIOR-NODES(W, m)                    ▷ recenter
6:         t = t + 1
7:     until t = T_max or no change in c and m
8:     return m, c
```

Algorithm 5.4 Bellman-Ford algorithm to compute distance and index of closest center. See Table 5.2 for variable definitions.

```
1: function BELLMAN-FORD(W, c)
2:     d_i ← ∞ for all i = 1, ..., N_node                ▷ initial distance
3:     m_i ← 0 for all i = 1, ..., N_node                ▷ initial membership undefined
4:     for a ← 1, ..., N_cluster do
5:         i ← c_a                                       ▷ cluster a has center node i
6:         d_i ← 0                                       ▷ distance of a center node to itself is zero
7:         m_i ← a                                       ▷ center node i belongs to its own cluster
8:     repeat
9:         done ← true
10:        for i, j such that W_{i,j} > 0 do             ▷ all pairs of adjacent nodes
11:            if d_j + W_{i,j} < d_i then               ▷ found a shorter distance to node j's center
12:                m_j ← m_i                             ▷ switch node j to the same cluster as i
13:                d_j ← d_i + W_{i,j}                   ▷ use the shorter distance via node i
14:                done ← false                          ▷ change was made; do not terminate
15:    until done
16:    return m, d
```

5.2.3.1 Theoretical observations

A significant advantage of standard Lloyd clustering, as in Algorithm 5.3, is the dependence on *off-the-shelf* algorithms such as Bellman-Ford. This allows us to establish key properties that will carry over to more advanced algorithms in the next section.

To begin, we note that standard Bellman-Ford terminates (in Theorem 5.1), an important property to maintain as we seek more balanced clusters.

Theorem 5.1. *Algorithm 5.4 terminates.*

Proof. This is a standard result [8, Section 8.7]. □

Likewise, while we assume the initial graph is connected, Definition 5.1 requires each of the clusters to be connected. Bellman-Ford provides this, as summarized in Theorem 5.2.

Theorem 5.2. *The clusters returned by Algorithm 5.4 are connected.*

Algorithm 5.5 Find the most-interior node (furthest from boundary) for each cluster. See Table 5.2 for variable definitions.

```
1:  function MOST-INTERIOR-NODES(W, m)
2:      B ← {}                                          ▷ border nodes
3:      for i, j such that W_{i,j} > 0 do               ▷ all pairs of adjacent nodes
4:          if m_i ≠ m_j then                           ▷ are nodes i and j in different clusters?
5:              B ← B ∪ {i, j}                          ▷ if so, add both of them to the border set
6:      ·, d ← BELLMAN-FORD(W, B)                       ▷ d is distance from cluster borders
7:      for i ← 1, ..., N_node do
8:          a ← m_i                                     ▷ a is the cluster index for node i
9:          c_a ← i                                     ▷ assign the highest-index node as cluster center
10:     for i ← 1, ..., N_node do
11:         a ← m_i                                     ▷ a is the cluster index for node i
12:         j ← c_a                                     ▷ j is the current cluster center
13:         if d_i > d_j then                           ▷ is node i further from the border than j?
14:             c_a ← i                                 ▷ if so, node i is the new cluster center
15:     return c
```

Proof. This follows from the proof of Theorem 5.3, using only the first case in the proof corresponding to Line 9 in Algorithm 5.9. □

5.3 Balanced Lloyd clustering

Lloyd clustering in Section 5.2.3 enables the construction of a *variable* number of clusters, based on the initial seeding. Yet, the method can result in poor *quality* clusters (cf. Figure 5.1). As an example, consider a nearest-neighbor weight matrix W based on distance and on a 6×6 structured mesh. Figure 5.2 illustrates two common scenarios in standard Lloyd clustering. The first is the emergence of long, narrow clusters. This is, in part, due to the method of finding boundaries in MOST-INTERIOR-NODES; in this case, the entire cluster (left figure) is comprised of boundary nodes, leaving no opportunity to re-center. The second artifact of standard Lloyd is that of disparate cluster sizes. Here, we observe both large clusters and clusters of a single point (right figure). An immediate goal in the algorithms of this section is to address these two points.

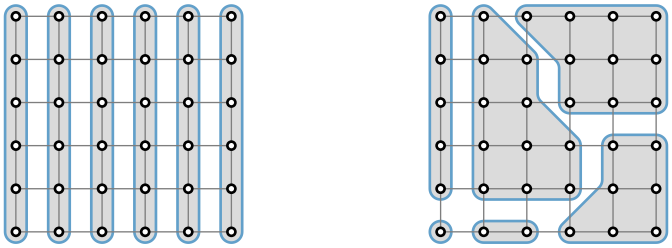


Figure 5.2: Two example clusterings from Lloyd clustering on a 6×6 mesh.

The Bellman-Ford algorithm is the central component of standard Lloyd, finding the shortest path for each seed and to find distal points for each boundary node in a cluster. In the following, we introduce a new method for centering nodes (see Algorithm 5.10), where we seek to minimize the total energy defined by

$$H = \sum_{i=1}^{N_{\text{node}}} d_i^2, \tag{5.1}$$

where d_i is defined to be the distance from node i to the center of the cluster for node i, namely c_a where cluster a satisfies $a = m_i$. This requires the computation of the shortest path for each pair of nodes in the cluster, a.k.a. the all-pairs shortest path problem. For this we turn to a per-cluster use of Floyd-Warshall [8] as detailed in Algorithm 5.6.

For cluster a, we note that the calculation of shortest paths in Algorithm 5.6 is $\mathcal{O}(s_a^3)$, where $s_a = |\{i \mid m_i = a\}|$ is the size of cluster a. In the following section, we establish linear complexity in the number of nodes, N_{node}, with assumptions on the maximum cluster size.

The introduction of energy as a target (see (5.1)) provides an opportunity to rebalance the clustering to account for small or large clusters. For this, we introduce a rebalancing algorithm that calculates the energy increase in splitting clusters and the energy decrease in eliminating clusters. The overall process relies on the distances from Floyd-Warshall. In Section 5.3.1, a balanced version of Bellman-Ford is introduced, leading to a balanced form of Lloyd clustering. The rebalancing algorithm is constructed in Section 5.3.2 followed by theoretical observations. The rebalanced Lloyd algorithm requires several components and we summarize the dependence in Figure 5.3.

Algorithm 5.6 Floyd-Warshall algorithm [8, Section 9.8] to find inter-node distances within each cluster. See Table 5.2 for variable definitions.

function CLUSTERED-FLOYD-WARSHALL(W, m)
  $V_a \leftarrow \{i \mid m_i = a\}$ for all $a = 1, \ldots, N_{\text{cluster}}$ ▷ nodes in cluster a
  for $a \leftarrow 1, \ldots, N_{\text{cluster}}$ **do**
    for $i, j \in V_a$ **do**
      $D_{i,j} \leftarrow \infty$ ▷ initial distance $i \to j$
      $P_{i,j} \leftarrow 0$ ▷ initial predecessor node for $i \to j$
      if $W_{i,j} > 0$ **then**
        $D_{i,j} \leftarrow W_{i,j}$ ▷ adjacent nodes have the adjacency distance
        $P_{i,j} \leftarrow i$ ▷ the predecessor is the tail node for adjacent pairs
      if $i = j$ **then**
        $D_{i,i} \leftarrow 0$ ▷ nodes are distance zero from themselves
        $P_{i,i} \leftarrow i$ ▷ nodes are their own predecessors to themselves
    for $k \in V_a$ **do** ▷ potential intermediate node on the path $i \to j$
      for $i, j \in V_a$ **do** ▷ all other node pairs within the cluster
        if $D_{i,k} + D_{k,j} < D_{i,j}$ **then** ▷ $i \to k \to j$ shorter than $i \to j$
          $D_{i,j} \leftarrow D_{i,k} + D_{k,j}$ ▷ switch to the shorter distance
          $P_{i,j} \leftarrow P_{k,j}$ ▷ take the predecessor from $k \to j$
  return D, P

5.3.1 Balanced algorithms

One disadvantage of Lloyd clustering is that the clusters are not guaranteed (nor expected) to be uniformly sized. In many practical settings, a node is likely to have nearly the same distance to multiple centers. In this case, Lloyd clustering randomly assigns the node to a cluster; in contrast, balanced Lloyd clustering targets uniformly sized clusters, as described in Algorithm 5.7. In the balanced approach, if a node has the same distance to different centers, the node is assigned to a smaller cluster, leading to increased uniformity across clusters. Specifically, Floyd-Warshall (Algorithm 5.6) replaces balanced Bellman-Ford (see Algorithm 5.9) to compute the centroid of each cluster. A node of a cluster having the minimum sum of squared distance to other cluster nodes is taken as centroid of that region (Algorithm 5.10). Consequently, long, narrow clusters as in Figure 5.2 will expose centers near the true center, whereas the boundary-distances used in standard Lloyd clustering leave any boundary centers

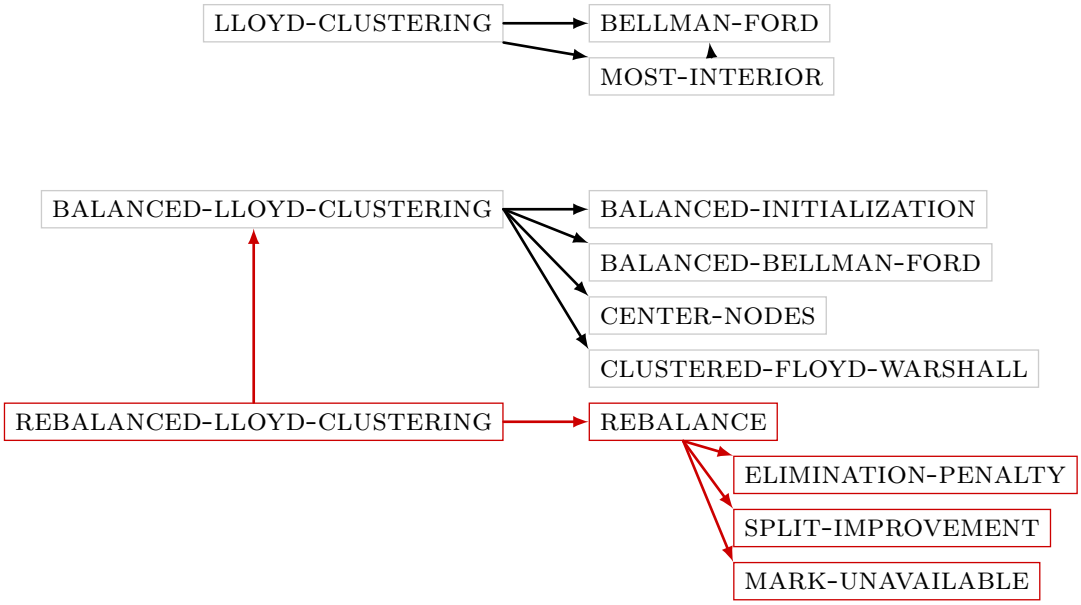


Figure 5.3: Algorithm dependence. Rebalancing components are highlighted in red.

unchanged.

Algorithm 5.7 Balanced version of Lloyd clustering. See Table 5.2 for variable definitions.

1: **function** BALANCED-LLOYD-CLUSTERING($W, c, T_{\max}, T_{\text{BFmax}}$)
2: $m, d, p, n, s \leftarrow$ BALANCED-INITIALIZATION(c, N_{node})
3: $t = 0$
4: **repeat**
5: $m, d, p, n, s \leftarrow$ BALANCED-BELLMAN-FORD($W, m, c, d, p, n, s, T_{\text{BFmax}}$)
6: $D, P \leftarrow$ CLUSTERED-FLOYD-WARSHALL(W, m)
7: $c, d, p, n \leftarrow$ CENTER-NODES(W, m, c, d, p, n, D, P)
8: $t = t + 1$
9: **until** $t = T_{\max}$ or no change in any of m, c, d, p, n, s
10: **return** m, c, d, p, n, s, D, P

Note that on Line 11 of Algorithm 5.9, the condition $d_i + W_{i,j} = d_j$ should be implemented using an approximate comparison if floating point arithmetic is being used.

Algorithm 5.8 Initialization for balanced algorithms. See Table 5.2 for variable definitions.

1: **function** BALANCED-INITIALIZATION(c, N_{node})
2: $m_i \leftarrow 0$ for all $i = 1, \ldots, N_{\text{node}}$ ▷ cluster membership for node i
3: $d_i \leftarrow \infty$ for all $i = 1, \ldots, N_{\text{node}}$ ▷ distance to node i from its cluster center
4: $p_i \leftarrow 0$ for all $i = 1, \ldots, N_{\text{node}}$ ▷ predecessor node for node i
5: $n_i \leftarrow 0$ for all $i = 1, \ldots, N_{\text{node}}$ ▷ number of predecessor nodes for node i
6: $s_a \leftarrow 1$ for all $a = 1, \ldots, N_{\text{cluster}}$ ▷ size of cluster a
7: **for** $a \leftarrow 1, \ldots, N_{\text{cluster}}$ **do**
8: $i \leftarrow c_a$ ▷ i is the center node index for cluster a
9: $d_i \leftarrow 0$ ▷ distance of center to node i from itself is zero
10: $m_i \leftarrow a$ ▷ center node i belongs to its own cluster
11: $p_i \leftarrow i$ ▷ centers are their own predecessors
12: $n_i \leftarrow 1$ ▷ centers have one predecessor
13: **return** m, d, p, n, s

5.3.2 Rebalancing clustering

Balanced Lloyd clustering improves the uniformity and roundness of the clusters in the Lloyd clustering. Yet there there is no guarantee that the energy is minimized *across* clusters, due to the initial seeding. In this section, we develop a rebalancing algorithm that reduces the overall energy in the clustering by *splitting* clusters into two and reducing energy, and by eliminating clusters leading to an increase in energy. The trade-off maintains a constant number of clusters, but reduces the total energy in the clustering. The rebalanced Lloyd algorithm is given in Algorithm 5.11 and the rebalancing algorithm itself is given in Algorithm 5.12.

The algorithm relies on two calculations, the first being in Algorithm 5.13, which iterates through each cluster and calculates the energy penalty (increase) resulting from eliminating a cluster and merging each node with its nearest cluster. The nearest cluster of a node is defined based on the distance of the centre of the cluster from the node. Similarly, Algorithm 5.14 computes the energy improvement (decrease) from optimally splitting each cluster into two clusters. Here, we determine the splitting (of each cluster) that results in the lowest energy by considering all possible pairs of new centers within the cluster.

With measures on the penalties and improvements in energy, the rebalancing

Algorithm 5.9 Balanced version of Bellman-Ford. See Table 5.2 for variable definitions.

```
1: function BALANCED-BELLMAN-FORD(W, m, c, d, p, n, s, T_BFmax)
2:     t ← 0; z^(t) ← z for all variables z                    ▷ only for use in proofs
3:     repeat
4:         done ← true
5:         for i, j such that W_{i,j} > 0 do                   ▷ all pairs of adjacent nodes
6:             𝔰_i ← s_{m_i} if m_i > 0, else 0                ▷ size of cluster containing node i
7:             𝔰_j ← s_{m_j} if m_j > 0, else 0                ▷ size of cluster containing node j
8:             switch ← false
9:             if d_i + W_{i,j} < d_j then                     ▷ j is closer to i's center than its own
10:                switch ← true
11:            if d_i + W_{i,j} = d_j then                     ▷ distance to j is similar from i's center
12:                if 𝔰_i + 1 < 𝔰_j then                       ▷ node i's cluster is smaller (by 2 or more)
13:                    if n_j = 0 then                         ▷ node j is free to switch (not a predecessor)
14:                        switch ← true
15:            if switch then
16:                s_{m_i} ← 𝔰_i + 1, s_{m_j} ← 𝔰_j − 1          ▷ update cluster sizes
17:                m_j ← m_i                                   ▷ switch node j to the same cluster as i
18:                d_j ← d_i + W_{i,j}                         ▷ use the distance via node i
19:                n_i ← n_i + 1, n_{p_j} ← n_{p_j} − 1        ▷ update predecessor counts
20:                p_j ← i                                     ▷ predecessor of node j is now i
21:                done ← false                                ▷ change was made; do not terminate
22:            t ← t + 1; z^(t) ← z for all variables z        ▷ only for use in proofs
23:    until t = T_BFmax or done
24:    T ← t                                                   ▷ only for use in proofs
25:    return m, d, p, n, s
```

algorithm proceeds by eliminating and splitting clusters in pairs, thereby reducing the total energy while keeping the number of clusters constant. At first, it eliminates the cluster with the smallest elimination penalty and splits the cluster with the largest split improvement, if these are distinct clusters. It then proceeds to eliminate the cluster with the second-smallest penalty and split the one with the second-largest improvement, again assuming they are distinct. This process continues until the energy will no longer be decreased (i.e., the next elimination penalty would be greater

Algorithm 5.10 Update center nodes to be the cluster centroids. See Table 5.2 for variable definitions.

function CENTER-NODES(W, m, c, d, p, n, D, P)
  $V_a \leftarrow \{i \mid m_i = a\}$ for all $a = 1, \ldots, N_{\text{cluster}}$ ▷ nodes in cluster a
  for $a = 1, \ldots, N_{\text{cluster}}$ **do** ▷ treat each cluster a separately
    for $i \in V_a$ **do**
      $q_i \leftarrow \sum_{j \in V_a} (D_{i,j})^2$ ▷ sum of squared distances to other cluster nodes
    $i \leftarrow c_a$ ▷ current cluster center
    for $j \in V_a$ **do** ▷ test node j as a new cluster center
      if $q_j < q_i$ **then** ▷ does node j have a *strictly* better metric?
        $i \leftarrow j$ ▷ j will be the new cluster center
    if $i \neq c_a$ **then** ▷ have we found a new center?
      $c_a \leftarrow i$
      $n_j \leftarrow 0$ for all $j \in V_a$ ▷ reset predecessor counts
      for $j \in V_a$ **do** ▷ update data for all nodes in the cluster
        $d_j \leftarrow D_{i,j}$
        $p_j \leftarrow P_{i,j}$
        $n_{p_j} \leftarrow n_{p_j} + 1$
  return c, d, p, n

Algorithm 5.11 Rebalanced version of Lloyd clustering. See Table 5.2 for variable definitions.

function REBALANCED-LLOYD-CLUSTERING($W, c, T_{\max}, T_{\text{BFmax}}$)
  $t = 0$
  repeat
    $m, c, d, p, n, s, D, P \leftarrow$ BALANCED-LLOYD-CLUSTERING($W, c, T_{\max}, T_{\text{BFmax}}$)
    $c \leftarrow$ REBALANCE(W, m, c, d, p, D)
    $t = t + 1$
  until $t = T_{\max}$ or no change in c
  return m, c, d, p, n, s, D, P

than or equal to the next split improvement), at which point rebalancing terminates. To access the clusters in sorted order we use an ARGSORT(L) function that returns the array of indexes $[i_1, i_2, \ldots]$ so that $L_{i_1}, L_{i_2}, \ldots$ will be in sorted order.

During rebalancing, we assume that the elimination penalties and split improvements of the clusters do not change as we are actually eliminating and splitting other clusters. However, the penalty and improvement of cluster a depend on its neighboring clusters. For this reason, when we eliminate or split a cluster, we mark all of its neighbors as unavailable for being eliminated or split themselves. This ensures that the penalties and improvement values remain correct for all clusters that are under consideration for elimination or splitting at each step.

Algorithm 5.12 Rebalance clusters by eliminating low-energy clusters and splitting the same number of high-energy clusters in two. See Table 5.2 for variable definitions.

function REBALANCE(W, m, c, d, p, D)
  $V_a \leftarrow \{i \mid m_i = a\}$ for all $a = 1, \ldots, N_{\text{cluster}}$ ▷ nodes in cluster a
  $L \leftarrow$ ELIMINATION-PENALTY(W, m, d, D)
  $(S, c^1, c^2) \leftarrow$ SPLIT-IMPROVEMENT(m, d, D)
  $M_a \leftarrow$ TRUE for all $a \leftarrow 1, \ldots, N_{\text{cluster}}$ ▷ all clusters are modifiable
  $L^{\text{sort}} \leftarrow$ ARGSORT(L)
  $S^{\text{sort}} \leftarrow$ ARGSORT(S)
  $i_{\text{L}} \leftarrow 1$ ▷ sorted index of cluster to eliminate
  $i_{\text{S}} \leftarrow N_{\text{cluster}}$ ▷ sorted index of cluster to split
  while $i_{\text{L}} \leq N_{\text{cluster}}$ and $i_{\text{S}} \geq 1$ **do**
    $a_{\text{L}} \leftarrow L^{\text{sort}}_{i_{\text{L}}}$ ▷ cluster to eliminate
    $a_{\text{S}} \leftarrow S^{\text{sort}}_{i_{\text{S}}}$ ▷ cluster to split
    if not $M_{a_{\text{L}}}$ or $a_{\text{L}} = a_{\text{S}}$ **then** ▷ is cluster a_{L} modifiable and distinct?
      $i_{\text{L}} \leftarrow i_{\text{L}} + 1$
      continue
    if not $M_{a_{\text{S}}}$ **then** ▷ is cluster a_{S} modifiable?
      $i_{\text{S}} \leftarrow i_{\text{S}} - 1$
      continue
    if $L_{a_{\text{L}}} \geq S_{a_{\text{S}}}$ **then** ▷ will the energy *not decrease*?
      break
    MARK-UNAVAILABLE($a_L, M, W, V_{a_{\text{L}}}$)
    MARK-UNAVAILABLE($a_S, M, W, V_{a_{\text{S}}}$)
    $c_{a_{\text{L}}} \leftarrow c^1_{a_{\text{S}}}$ ▷ eliminate cluster a_{L}
    $c_{a_{\text{S}}} \leftarrow c^2_{a_{\text{S}}}$ ▷ split cluster a_{S}
  return c

Algorithm 5.13 Calculate the energy increase that would result from eliminating each cluster. See Table 5.2 for variable definitions.

```
1: function ELIMINATION-PENALTY(W, m, d, D)
2:     V_a ← {i | m_i = a} for all a = 1, ..., N_cluster        ▷ nodes in cluster a
3:     for a = 1, ..., N_cluster do
4:         L_a ← 0                                  ▷ energy penalty for eliminating cluster a
5:         for i ∈ V_a do
6:             d_min ← ∞                ▷ minimum distance to a different cluster center
7:             for j ∈ V_a do                              ▷ look for connectivity via j
8:                 for k such that W_{k,j} > 0 do                    ▷ all neighbors of j
9:                     if m_k ≠ m_j then                ▷ is k in a different cluster to j?
10:                        if d_k + W_{k,j} + D_{j,i} < d_min then      ▷ is k's center closer?
11:                            d_min ← d_k + W_{k,j} + D_{j,i}
12:            L_a ← L_a + (d_min)^2                          ▷ add the new energy for i
13:        L_a ← L_a − Σ_{i∈V_a} (d_i)^2                  ▷ subtract the current energy metric
14:    return L
```

5.3.3 Theoretical observations

We have formulated the balanced Bellman-Ford algorithm with a cap on the maximum number of iterations, which will be necessary for proving linear complexity in Theorem 5.6. In practice T_{BFmax} can be chosen to be the maximum expected cluster radius and implementations should warn if Algorithm 5.9 reaches this limit, as this may indicate that the clusters are not connected. This observation relies on the following result.

Theorem 5.3. *The clusters returned by Algorithm 5.9 are connected if it terminates before the maximum number of iterations.*

Proof. The proof relies on understanding the state of the variables within Algorithm 5.9 as the algorithm iterates. We denote this by using a superscript, $z^{(t)}$ to indicate the state of variable z at a given "time", t, in the algorithm, with time T denoting completion.

We wish to show that, for each j, $m_j^{(T)} = m_i^{(T)}$ where $i = p_j^{(T)}$, which ensures that the predecessor-paths to cluster centers are contained within each cluster. Suppose

Algorithm 5.14 Calculate the energy decrease that would result from optimally splitting each cluster in two. See Table 5.2 for variable definitions.

```
1: function SPLIT-IMPROVEMENT(m, d, D)
2:     V_a ← {i | m_i = a} for all a = 1, ..., N_cluster        ▷ nodes in cluster a
3:     for a = 1, ..., N_cluster do
4:         S_a ← ∞                          ▷ energy improvement for splitting cluster a
5:         for i ∈ V_a do                                      ▷ first possible new center
6:             for j ∈ V_a do                                  ▷ second possible new center
7:                 S_new ← 0                                   ▷ energy with centers i and j
8:                 for k ∈ V_a do                              ▷ compute cost for node k
9:                     if D_{i,k} < D_{j,k} then               ▷ is k closer to center i or j?
10:                        S_new ← S_new + (D_{i,k})^2
11:                    else
12:                        S_new ← S_new + (D_{j,k})^2
13:                if S_new < S_a then                         ▷ is this a better split?
14:                    S_a ← S_new                             ▷ store the new energy
15:                    c_a^1 ← i                               ▷ store the new centers i and j
16:                    c_a^2 ← j
17:        S_a ← Σ_{i∈V_a} (d_i)^2 − S_a          ▷ improvement from current cluster energy
18:    return S, c^1, c^2
```

Algorithm 5.15 Mark a cluster and all of its neighbors as unavailable. See Table 5.2 for variable definitions.

```
1: function MARK-UNAVAILABLE(a, M, W, V_a)
2:     M_a ← False                                             ▷ cluster a is unavailable
3:     for i ∈ V_a do
4:         for j such that W_{i,j} > 0 do                      ▷ all neighboring nodes of cluster a
5:             M_{m_j} ← False                                 ▷ cluster of node j is unavailable
```

not and let t be the last iteration when m_j and p_j were updated by Lines 17 and 20. Taking $i = p_j^{(t)}$ we have $m_i^{(t)} = m_j^{(t)} = m_j^{(T)} \neq m_i^{(T)}$ so there must be a later $t' > t$ at which m_i was updated for the last time. At this later time, we must have at least one of the following cases.

Case 1: condition on Line 9 is true. Then $d_i^{(t')} < d_i^{(t)}$, and so $d_i^{(t')} + W_{i,j} <$

$d_i^{(t)} + W_{i,j} = d_j^{(t)} = d_j^{(T)}$. This is a contradiction because we cannot have $d_j > d_i + W_{i,j}$ when the algorithm terminates.

Case 2: conditions on Lines 11 and 13 are true. Then $n_i^{(t')} = 0$, which is impossible since $p_j^{(t')} = p_j^{(t)} = i$. □

To understand the behavior of the balanced algorithms, we consider a modified energy that includes a second term for the cluster sizes. Define

$$H_\delta = \sum_{i=1}^{N_{\text{node}}} (d_i)^2 + \delta \sum_{a=1}^{N_{\text{cluster}}} (s_a)^2, \tag{5.2}$$

where d_i is the distance from node i to its cluster center, $s_a = |\{i \mid m_i = a\}|$ is the size (number of nodes) of cluster a, and

$$\delta = \left(\frac{\Delta_{\min}}{N_{\text{node}}} \right)^2 \tag{5.3}$$

is chosen based on the minimum difference, $\Delta_{\min}$, between distinct values of $W_{i,j}$ (or an arbitrarily small positive number if there are no distinct values of $W_{i,j}$). The first term in (5.2) is the sum of squared distances from nodes to their cluster centers, while the second term is the sum of squared cluster sizes. Note that δ is chosen so that the second term in (5.2) is always less than the minimum possible increment in the first term.

Lemma 5.1. *Algorithm 5.9 results in a decrease of the energy* (5.2)*, or preserves the energy if no change is made to the clustering.*

Proof. We will show that all steps in the algorithm that change d_j or s_a result in a strict decrease of H_δ.

Case 1: updates by Lines 16 and 18 with d_j strictly decreasing. Then the reduction in the first term in H_δ is at least $(\Delta_{\min})^2$ and any increase in the second term in H_δ is less than $(N_{\text{node}})^2$, so the definition of δ means the decrease strictly dominates.

Case 2: updates by Lines 16 and 18 with d_j constant and $\mathfrak{s}_j > \mathfrak{s}_i + 1$. Then the first term in H_δ is constant and $s_{m_i} \leftarrow \mathfrak{s}_i + 1$ and $s_{m_j} \leftarrow \mathfrak{s}_j - 1$ results in a strict decrease of $(s_{m_i})^2 + (s_{m_j})^2$. □

Lemma 5.2. *Algorithm 5.10 results in a decrease of the energy (5.2), or preserves the energy if no change is made to the clustering.*

Proof. Only updates by Line 14 will change d_j. Because the change is caused by the use of j as the new center, and $q_j < q_i$, the first term in H_δ strictly decreases and the second term is unchanged. □

Theorem 5.4. *Algorithm 5.7 terminates, even if $T_{max} = \infty$.*

Proof. From Lemmas 5.1 and 5.2, all steps in the algorithms that change d_i or s_a result in a strict decrease of H_δ. Because H_δ is positive and can only take a finite number of values, and we terminate when no changes are made, this ensures termination. □

Theorem 5.5. *Algorithm 5.12 results in a decrease or preservation of the energy (5.2).*

Proof. Because of Line 19, each elimination/split pairing explicitly results in a decrease of the first term in (5.2) and thus also a decrease in the overall value of H_δ due to the choice of δ. □

To give bounds on the computational complexity of the algorithms we require the following assumptions on the graph structure.

Assumption 5.3.1. *Assume that the number of edges in G incident on each vertex in the graph is bounded independently of N_{node}, and that the initial centers, c, are such that clusters found by Algorithm 5.9 have size bounded independently of N_{node}.*

Theorem 5.6. *Under Assumption 5.3.1, if T_{max} and T_{BFmax} are both bounded independently of N_{node}, then the total cost of Algorithm 5.7 is $\mathcal{O}(N_{node})$.*

Proof. This follows from cost estimates for each of the components of Algorithm 5.7. The inner loop of Algorithm 5.9 has complexity equal to the number of edges in G, which is $\mathcal{O}(N_{\text{node}})$ by Assumption 5.3.1. If $T_{\text{BFmax}} = \mathcal{O}(1)$, then Algorithm 5.9 has complexity $\mathcal{O}(N_{\text{node}})$. The cost of Floyd-Warshall on each cluster is cubic in the cluster size, which we assume (as a function of the initial centers and clusters) to be $\mathcal{O}(1)$, giving a total cost of $\mathcal{O}(N_{\text{cluster}}) = \mathcal{O}(N_{\text{node}})$. Algorithm 5.10 also has linear complexity. Since T_{max} in Algorithm 5.7 is also $\mathcal{O}(1)$, the total complexity is, then, $\mathcal{O}(N_{\text{node}})$. □

Theorem 5.7. *Under Assumption 5.3.1, Algorithm 5.12 terminates with cost at most $\mathcal{O}(N_{\text{node}} \log N_{\text{node}})$ and, if T_{max} is bounded independently of N_{node}, then Algorithm 5.11 also has $\mathcal{O}(N_{\text{node}} \log N_{\text{node}})$ total cost.*

Proof. From Theorem 5.6, the cost of Algorithm 5.7 is $\mathcal{O}(N_{\text{node}})$. Both Algorithm 5.13 and Algorithm 5.14 iterate over all clusters and perform bounded work per cluster (using the bound on cluster size from Assumption 5.3.1), so they have cost $\mathcal{O}(N_{\text{cluster}}) = \mathcal{O}(N_{\text{node}})$. Similarly, Algorithm 5.15 has cost independent of N_{node} because it only iterates over nodes within a single cluster. The cost of Algorithm 5.12 is thus $\mathcal{O}(N_{\text{node}} \log N_{\text{node}})$ because Lines 6 and 7 are $\mathcal{O}(N_{\text{cluster}} \log N_{\text{cluster}})$ and all other loops and subroutines are $\mathcal{O}(N_{\text{node}})$. Finally, assuming $T_{\max} = \mathcal{O}(1)$, we have the same cost for Algorithm 5.11. □

Remark 5.1. *The algorithmic complexity of Algorithm 5.12 and, hence, that of Algorithm 5.11, can be reduced to $\mathcal{O}(N_{\text{node}})$ by changing the algorithm to separately treat fixed-size sets of clusters. That is, rather than considering all clusters at once, partition the set of clusters into subsets and run Algorithm 5.12 separately on each subset. This will avoid the $\mathcal{O}(N_{\text{node}} \log N_{\text{node}})$ sorts in Lines 6 and 7, leaving the cost as linear in N_{node}. We expect this would lead to some slight reduction in the quality of the rebalance, because eliminate/split pairings will only be considered within a subset but, for large subsets, we would not expect this to make a significant difference. This subset approach is also the natural way to parallelize Algorithm 5.12, with one subset per processor.*

Remark 5.2. *Parallelization of Algorithms 5.7 and 5.11 relies on parallelization of the other underlying algorithms. Both Algorithms 5.6 and 5.10 operate independently on each cluster and are, thus, naturally parallelizable. Algorithm 5.9 could be naturally parallelized by applying it independently to the set of nodes owned by each processor in a parallel decomposition.*

5.4 Numerical Results

In this section, we highlight the value of balanced Lloyd clustering with rebalancing for smoothed aggregation multigrid. All computations are performed with PyAMG [3]. Unless stated otherwise, all results below consider a standard Poisson problem of form

$$-\nabla \cdot \nabla U = F \quad \text{in } \Omega, \tag{5.4a}$$

$$\boldsymbol{n} \cdot \nabla U = 0 \quad \text{on } \partial\Omega, \tag{5.4b}$$

where Neumann boundary conditions are used to highlight clustering near the boundary.[3] (5.4) is discretized using either standard P^1 linear finite elements on a triangulation of the domain, Ω, or Q^1 bilinear finite elements on a quadrilateral mesh of Ω, yielding a matrix problem of the form

$$Au = f. \tag{5.5}$$

In the following convergence tests, f is set to zero and a random approximation to u is used to initialize the AMG cycling.

We consider three main cases of clustering in the context of AMG: standard Lloyd clustering (Algorithm 5.3), balanced Lloyd clustering (Algorithm 5.7), and balanced Lloyd clustering with rebalancing (Algorithm 5.12). For each of these, we require a definition of the weight matrix, W, and the number of clusters, N_{cluster}. In each case, we bound the number of inner iterations of Lloyd clustering at five and the number of rebalance sweeps at four; in practice, this is a conservative bound and the iterations complete much earlier (due to no change in the clustering state).

To form the weight matrix, W, we consider the so-called *evolution* measure [18] which associates a value of strength for each edge in the graph of A in (5.5) based on smoothing properties. This leads to an initial non-negative weight matrix, $\widehat{W}$, where a large edge value $\widehat{W}_{i,j}$ indicates that nodes i and j should be clustered together. The algorithms above make use of an assumption that the graph associated with W is connected, but this is not guaranteed to be the case for that of $\widehat{W}_{i,j}$. Thus, we augment $\widehat{W}$ with a small padding for each edge in A, defining $\widetilde{W}$ as

$$\widetilde{W}_{i,j} \leftarrow \widehat{W}_{i,j} + 0.1 \quad \text{if } A_{i,j} \neq 0 \tag{5.6}$$

The Lloyd-based clustering presented here is based on shortest distances in the graph of W. As a result, we consider the inverse strength as a proxy for distance. This results in defining W so that

$$W_{i,j} = \frac{1}{\widetilde{W}_{i,j}} \text{ if } \widetilde{W}_{i,j} \neq 0, \tag{5.7}$$

so that strong edges refer to shorter distances. With this inversion, the additional padding added above indicates a long distance in the weight matrix.

[3] With Dirichlet conditions, we would observe "singleton" clusters for the isolated points. This does not impact the method, only visualization.

5.4.1 Varying cluster numbers

While Greedy and MIS-based clustering have been used successfully in many settings, they do not provide a mechanism to control the number of resulting clusters. Here, we explore the ability of Lloyd clustering to target specific numbers of clusters. As motivation, consider the model problem on an unstructured triangulation of the unit disk with 10 245 vertices and 20 158 elements. We construct multigrid hierarchies using rebalanced Lloyd clustering, setting the target number of points in each cluster at each level to a fixed value between 3 and 20. We estimate the asymptotic convergence factor by the geometric mean of the last five residual norms at convergence, say k iterations:

$$\rho = \left(\frac{\|r^{(k)}\|}{\|r^{(k-5)}\|} \right)^{\frac{1}{4}}, \tag{5.8}$$

where $r^{(k)} = f - Au^{(k)}$ is the residual vector after k iterations. Combined with a model for the *cost* of each multigrid cycle, given by the total number of non-zeros in the sparse-matrix operations in the cycle (i.e., the *cycle complexity*, χ), this leads to a measure of the work per digit of accuracy (WPD) for the method:

$$\text{WPD} = \frac{\chi}{-\log_{10}(\rho)}. \tag{5.9}$$

Figure 5.4 shows that the efficiency (and effectiveness) of an AMG method can vary depending on the (average) number of points per cluster; in this case, we observe that very small clusters lead to rapid convergence (small ρ), yet due to the slower coarsening, the total complexity of the multigrid cycle is higher.

In the end, balanced Lloyd clustering with rebalancing leads to well-formed clusters and the ability to use a vast range of cluster sizes. Figure 5.5 illustrates a range of cases for a smaller mesh of the same domain, from five (large) clusters at one extreme to 250 small clusters including singleton and many pairwise clusters. True pairwise clustering [5, 16] is not represented; however, it remains an open question whether a Lloyd-type algorithm could render nearly pairwise clustering using modified criteria for tiebreaking and rebalancing.

5.4.2 Tiebreaking

Algorithm 5.9 introduces "tiebreaking" on Line 12. If a node in the graph is equidistant from multiple centers, then the node becomes a member of the neighboring cluster if the neighboring cluster is smaller by two in size than the current cluster of

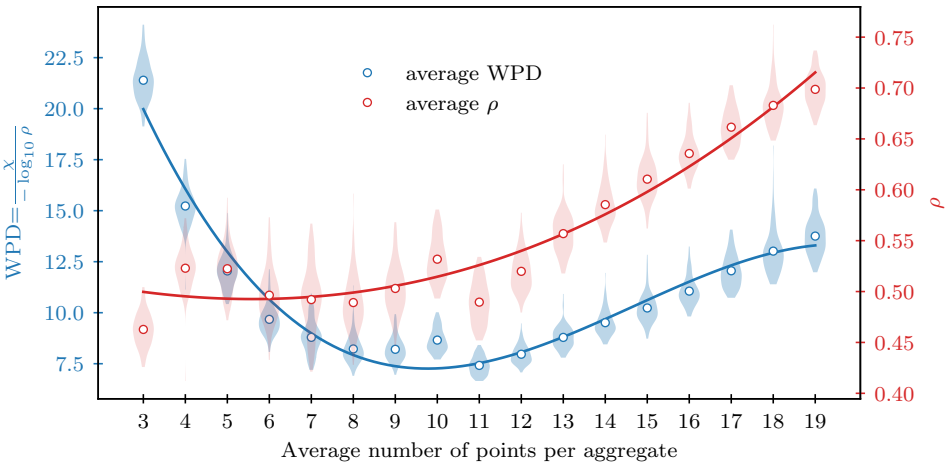


Figure 5.4: Work per digit (WPD) of accuracy and convergence ρ for clustering sizes ranging from 3–19 points per cluster (on average) using rebalanced Lloyd clustering. The average over 100 runs is marked $\circ$ and a trendline from a smoothed cubic spline is given for the mean (solid).

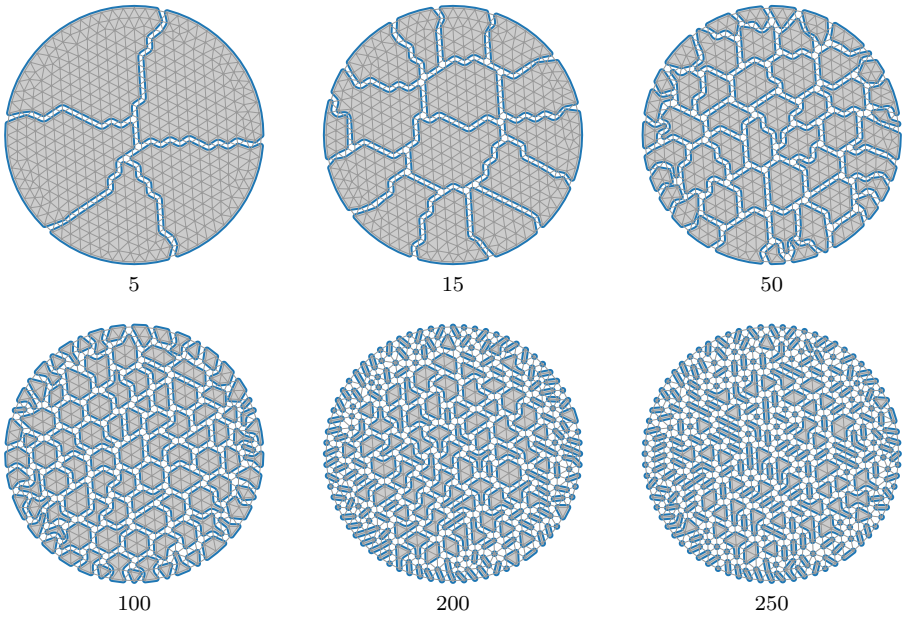


Figure 5.5: Example clustering patterns with the number of clusters ranging from 5 to 250 using rebalanced Lloyd clustering.

the node. Tiebreaking in the balanced Bellman-Ford algorithm impacts the uniformity of the sizes of the clusters. To quantify uniformity, we consider discretizing (5.4) on a uniform 64×64 quadrilateral mesh. We cluster the nodes using the balanced Bellman-Ford algorithm *with* and *without* tiebreaking, requesting the number of clusters be equal to 10% of the fine-grid number of nodes (rounded down when this is not an integer). We randomly distribute the initial seeding 1000 times and, in each case, compute the following metrics: the number of zero diameter clusters (i.e., singleton clusters), the standard deviation in the number of nodes per cluster, and the energy for each clustering (defined by (5.1)).

Figure 5.6 shows the number of clusters having zero diameter with and without tiebreaking, highlighting that tiebreaking substantially decreases the number of clusterings with zero diameter clusters, from over one-third of clusterings to about one percent. Likewise, Figure 5.7 (left) shows the effect of tiebreaking on the distribution of the standard deviation in the number of nodes. Here, tiebreaking leads to a decrease yielding clusters more uniform in size. Tiebreaking also contributes to clusters that are more round — this is supported by Figure 5.7 (right), where we see that tiebreaking decreases the energy of the system. We emphasize that tiebreaking is an inexpensive strategy that clearly improves performance of the clustering.

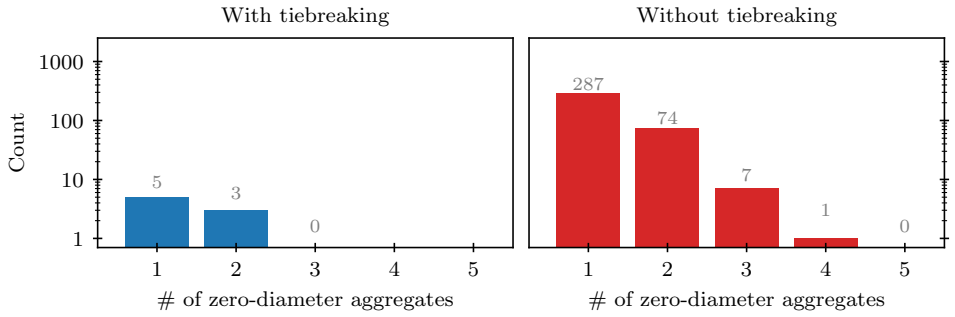


Figure 5.6: Distribution of the number of clusters having zero diameter for balanced Lloyd clustering with or without tiebreaking for a 64×64 mesh.

5.4.3 Rebalancing

To quantify the improvements in cluster quality as we move from standard to balanced, and then to rebalanced Lloyd clustering, we again consider a 64×64 quadrilateral mesh. Nodes in this mesh are clustered using the three methods, again using

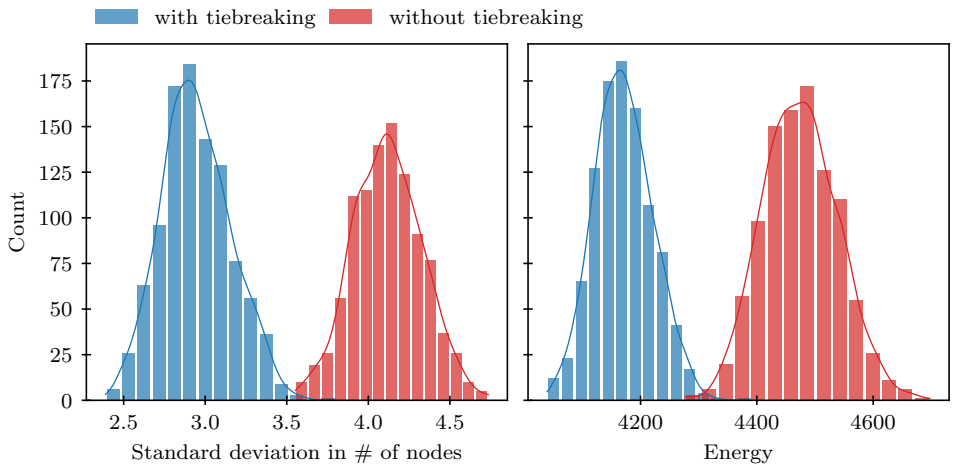


Figure 5.7: Distribution of the standard deviation in the number of nodes and distribution of energy for balanced Lloyd clustering with or without tiebreaking on a 64×64 mesh.

10% of N_{node} to determine N_{cluster}. In each case, the clustering is repeated 1000 times, yielding a standard deviation of cluster diameter, standard deviation of number of nodes in clusters, and energy for each test. The results are averaged and the same experiment is performed for 16×16, 32×32, and 128×128 meshes.

Figure 5.8 shows the distributions for each method in the case of a 64×64 mesh. Lower standard deviation of diameter and standard deviation of number of nodes suggest that the clusters that result from rebalanced Lloyd are more uniform in shape and size compared to the other methods. This is also reflected by the lower energy for rebalanced Lloyd clustering. The figure also highlights that *variation* in the metrics is lower for rebalanced Lloyd, pointing to the consistency in the method over multiple runs.

Figure 5.9 shows the difference between the maximum and minimum diameters and the energy, averaged over 1000 samples, for each of the clustering methods as we vary problem size. The figures underscore that rebalanced Lloyd yields more uniform, rounded clusters having less energy than the other two clustering methods as the mesh size grows.

5.4.4 Algebraic multigrid convergence

In the application of clustering to algebraic multigrid, cluster quality plays an important role in overall convergence of the method, but one that is not yet quantified by

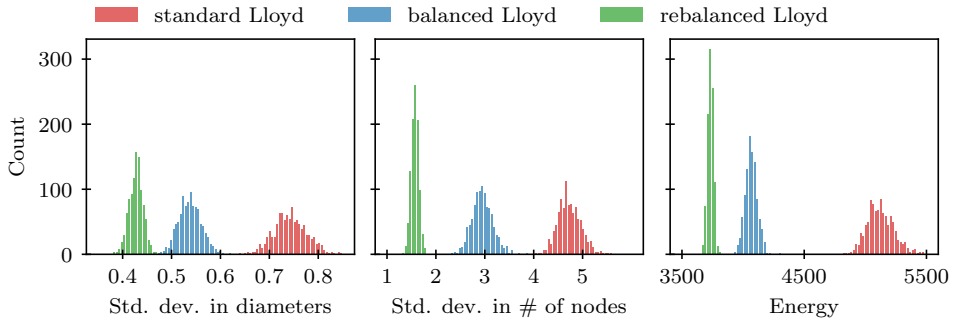


Figure 5.8: Distribution of standard deviation in diameters, distribution of standard deviation in number of nodes, and distribution in energy for different clustering methods for a 64×64 mesh.

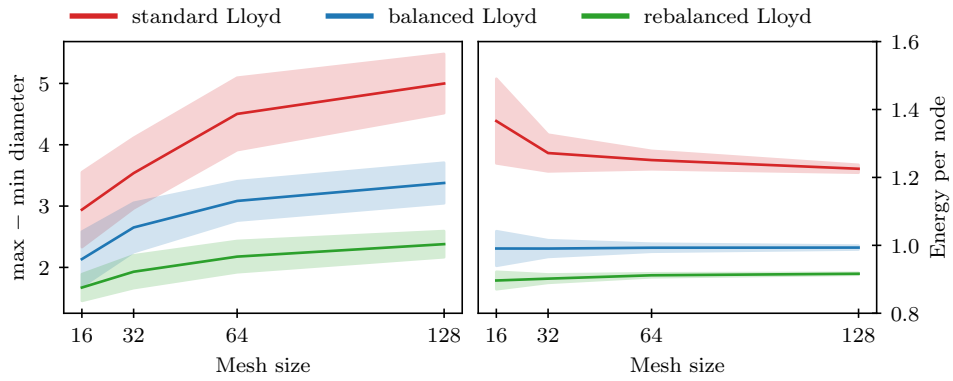


Figure 5.9: (Left) Difference between maximum and minimum diameters of clusters averaged over 1000 samples; (Right) Energy per node averaged over 1000 samples. The shaded regions mark one standard deviation from the mean.

existing sharp measures. While we can easily confirm improvement (or degradation) in the measured convergence factor after making a change to a clustering, it is difficult to directly assess if an individual cluster is the cause of poor convergence.

One way to *localize* a bound on AMG convergence is to consider the classical bound based on smoothing and approximation properties [13, 20]. This theory considers the convergence of a two-grid cycle with post-relaxation given by $u \leftarrow u + M(f - Au)$ and coarse-grid correction given by $u \leftarrow u + P(P^TAP)^{-1}P^T(f - Au)$. We write $G = I - MA$ and $T = I - P(P^TAP)^{-1}P^TA$ as the error-propagation operators of relaxation and coarse-grid correction, respectively, with the error-propagation operator of the two-grid scheme given by GT. The diagonal of SPD matrix A is denoted by D. In

what follows, we assume that A is SPD, P is of full rank, and $\|G\|_A < 1$. Theorem 4 of [13] shows that if there exist constants $\alpha, \beta > 0$ such that

$$\|Ge\|_A^2 \leq \|e\|_A^2 - \alpha\|e\|_{AD^{-1}A}^2 \text{ for all } e,$$
$$\text{and } \|Te\|_A^2 \leq \beta\|Te\|_{AD^{-1}A}^2 \text{ for all } e,$$

then $\|GT\|_A \leq (1 - \alpha/\beta)^{1/2}$. The first of these is known as the *smoothing property,* since it concerns the action of relaxation, G, on errors, e. The second is referred to as the *approximation property*, since it quantifies the action of the coarse-grid correction process. Equations (19) and (20) of [13] show that this approximation property is guaranteed by the existence of a constant $\beta > 0$ such that $\inf_{e_c} \|e - Pe_c\|_D^2 \leq \beta\|e\|_A^2$ for all e. Choosing $e_c = (P^T DP)^{-1} P^T De$ and defining $T_D = I - P(P^T DP)^{-1} P^T D$ then allows us to quantify such a β as

$$\beta = \sup_{e \neq 0} \frac{e^T T_D^T D T_D e}{e^T A e}. \tag{5.10}$$

We find this β by solving for the largest eigenvalue of the generalized eigenvalue problem $T_D^T D T_D e = \lambda A e$, and let e be the associated eigenvector. To localize the measure over a single cluster, we decompose the inner product in the numerator into a sum over clusters, writing $\beta = \sum_{i=1}^{N_{\text{cluster}}} \beta_i$, where

$$\beta_i = \frac{\left(\sum_{j \in \Omega_i} (DT_D e)_j (T_D e)_j\right)}{e^T A e} \tag{5.11}$$

This comes from writing the numerator of (5.10) as the inner product of $DT_D e$ with $T_D e$, and then localizing the summation in that inner product over each cluster.

We again consider the Poisson problem on a triangulation of the unit disk with 528 unknowns, and compute (5.11) for each cluster generated by each method. Figure 5.10 shows that the extreme values of β_i are reduced through rebalancing. Indeed, this is reflected in the convergence shown in Figure 5.11, where we observe a dramatic reduction in the number of iterations for solvers with these clusters. It is, of course, important to note that not *every* clustering generated by standard Lloyd exhibits similarly poor performance. The quality of the initial clustering used to seed the algorithm plays an important role in determining the multigrid performance. Results seen here are for a representative, randomly generated, initial seeding.

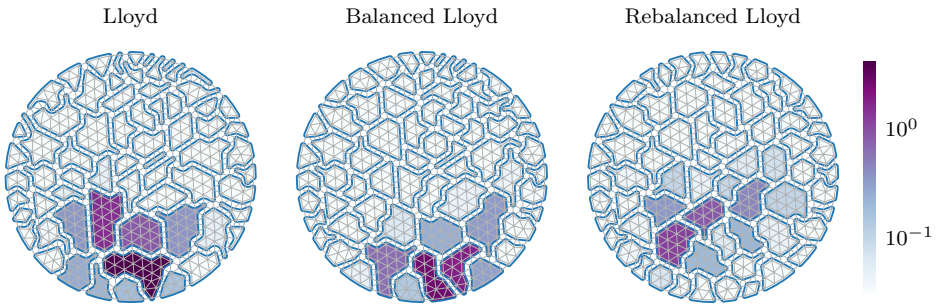


Figure 5.10: Localizing β to each cluster via (5.11).

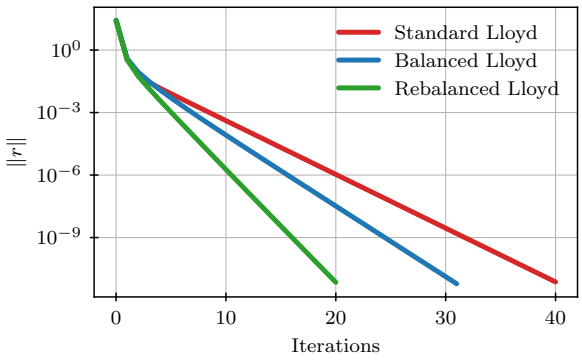


Figure 5.11: Example convergence for two-level AMG with different clusterings.

5.4.5 Additional problems in Algebraic Multigrid

As additional evidence of the effectiveness of rebalanced Lloyd clustering, we consider several examples in both 2D and 3D.

3D restricted channel: The 3D domain Ω is defined by a spline on the points

$$[(0, 4, -8), (0, 4, -6), (0, 1, 0), (0, 4, 6), (0, 4, 8)],$$

rotated about the z-axis, (see Table 5.1). A 3D tetrahedral mesh with 16 921 elements is generated with Gmsh [9] through pygmsh [21]. We use Firedrake [19] to discretize (5.4) with linear finite elements on tetrahedra, and select an average of 25 points per cluster.

2D restricted channel: The 2D domain Ω is defined by $[-2, 2] \times [-1, 1] \setminus C$ with $C = C^+ \cup C^-$, for $C^\pm$ representing discs of radius 0.8 at $(0, \pm 1)$ (see Table 5.1). As for the 3D restricted channel, we use Gmsh to generate a graded, triangular mesh with 5832 elements, with a characteristic length of 0.012 at the center and growing to 0.12 at the left/right edges. This forces tighter clustering toward the center, as

shown in Table 5.1. The discretization matrix for (5.4) is constructed with linear finite elements, and we target clusters of size 8.

2D anisotropic diffusion: The 2D domain is defined by the unit square, and we consider the problem $-\nabla \cdot K\nabla u = f$ with pure Dirchlet conditions. We define the anisotropic diffusion tensor as $K = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & \\ & \varepsilon \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}^T$, for $\varepsilon = 0.1$ and $\theta = \pi/3$. We discretize this on a 42×42 uniform mesh (with 1681 elements) and Q1 bilinear elements, and specify a target cluster size of 12.

P2 elements: The 2D domain is a unit disc, on which we consider (5.4). A triangular mesh is constructed with 982 elements and P2 quadratic finite elements are used to generate the discretization matrix. We specify 5 nodes per cluster.

In each of the examples of Table 5.1, a zero right-hand side is used to assess convergence of the smoothed aggregation multigrid solver. From the convergence histories, we see that rebalanced Lloyd clustering improves solver convergence, even for these relatively benign problems. For the restricted channel problems, the resulting clustering resembles the expected isotropic behavior with well rounded clusters. Likewise, in the case of anisotropy, we see that the clustering mimics the diffusion direction, while maintaining balance across clusters. Finally, the P2 case reveals the benefit of specifying the coarsening ratio: in this case, the coarsening ratio of 1/5 outperforms greedy coarsening (which yields a ratio of around 1/10).

As a final example, Table 5.1 highlights a parallel partitioning of an arc heated combustion channel at the University of Illinois Urbana-Champaign[4]. In this case, rebalanced Lloyd effectively partitions the ∼100k mesh elements, keeping refined features such as the injector local to a cluster.

5.5 Conclusions and extensions

In this paper, we study and extend the use of Lloyd's algorithm for determining clusters in graphs. Our proposed *balanced* and *rebalanced* Lloyd clustering algorithms are linear in time, guarantee connected clusters, and are consistent with minimizing a quadratic energy functional. In addition, the algorithms are implemented in Python/C++ and are available through the open source project PyAMG [3]. One major topic for future work is the choice of that energy functional; while the steps in the algorithms above are consistent with an ℓ^2-distance style energy, they can easily

[4]https://tonghun.mechse.illinois.edu/research/hypersonics-act-ii/, https://ceesd.illinois.edu/

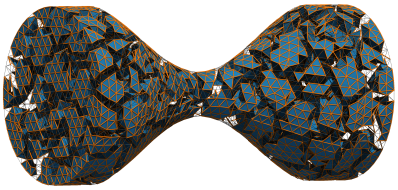
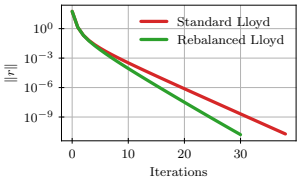
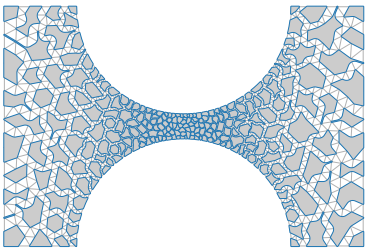
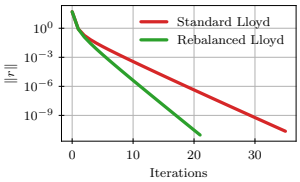
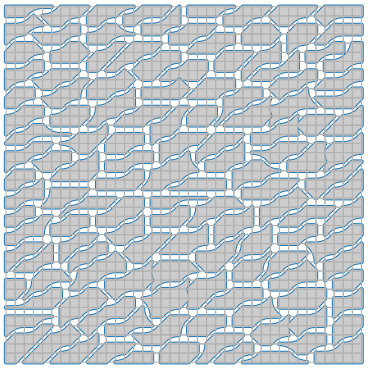
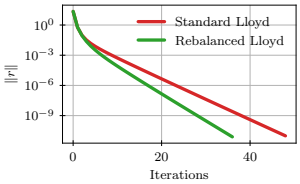
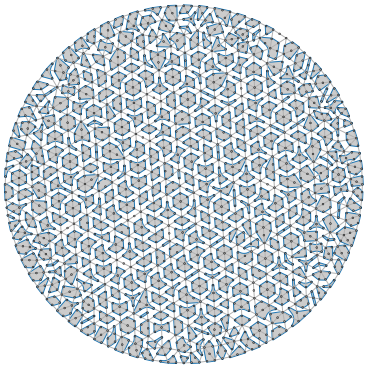
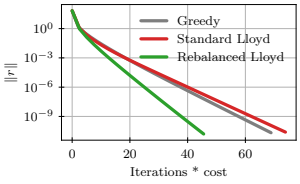
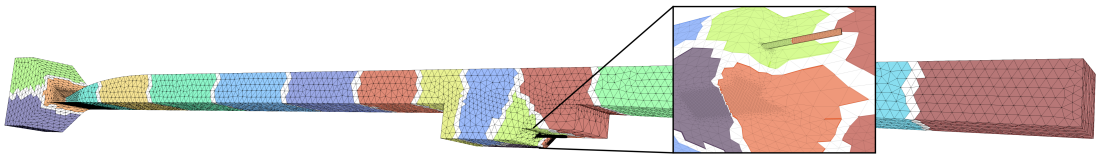
3D restricted channel		
2D restricted channel		
2D anisotropic diffusion		
P2 elements		
ACTII mesh		

Table 5.1: Additional examples. *ACTII mesh credit*: Mike Anderson at UIUC.

be extended to other energy functionals in a consistent way. Theoretical guidance is clearly needed to determine the proper choice of such a functional. We also note that we consider only serial algorithms in this paper; properly extending these approaches to their parallel counterparts is also an important subject for future research.

Acknowledgments

The work of S.P.M. was partially supported by an NSERC Discovery Grant. This work does not have any conflicts of interest.

Bibliography

[1] David Arthur and Sergei Vassilvitskii. K-Means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics. ISBN 9780898716245.

[2] Nathan Bell, Steven Dalton, and Luke N Olson. Exposing fine-grained parallelism in algebraic multigrid methods. *SIAM Journal on Scientific Computing*, 34(4): C123–C152, 2012.

[3] Nathan Bell, Luke N. Olson, and Jacob Schroder. PyAMG: Algebraic multigrid solvers in Python. *Journal of Open Source Software*, 7(72):4142, 2022. doi: 10.21105/joss.04142.

[4] W. N. Bell. *Algebraic multigrid for discrete differential forms*. PhD thesis, University of Illinois at Urbana-Champaign, 2008. URL `http://hdl.handle.net/2142/81820`.

[5] J. Brannick, Y. Chen, J. Kraus, and L. Zikatanov. Algebraic multilevel preconditioners for the graph laplacian based on matching in graphs. *SIAM Journal on Numerical Analysis*, 51(3):1805–1827, 2013. doi: 10.1137/120876083.

[6] Marian Brezina, Jan Mandel, et al. Convergence of algebraic multigrid based on smoothed aggregation. *Numerische Mathematik*, 88(3):559–579, 2001.

[7] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial*. SIAM Books, Philadelphia, 2000. Second edition.

[8] Jeff Erickson. Algorithms, 2019. URL `http://jeffe.cs.illinois.edu/teaching/algorithms/`.

[9] Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009. doi: https://doi.org/10.1002/nme.2579.

[10] George Karypis and Vipin Kumar. METIS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices, 2022. URL `https://github.com/KarypisLab/METIS`. v5.1.1.

[11] B. Kelley and S. Rajamanickam. Parallel, portable algorithms for distance-2 maximal independent set and graph coarsening. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 280–290, Los Alamitos, CA, USA, June 2022. IEEE Computer Society. doi: 10.1109/IPDPS53621.2022.00035.

[12] Stuart Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[13] S. MacLachlan and L. Olson. Theoretical bounds for algebraic multigrid performance: review and analysis. *Numer. Linear Alg. Appl.*, 21(2):194–220, 2014.

[14] Thomas A. Manteuffel, Luke N. Olson, Jacob B. Schroder, and Ben S. Southworth. A root-node-based algebraic multigrid method. *SIAM J. Sci. Comput.*, 39(5):S723–S756, 2017. ISSN 1064-8275. doi: 10.1137/16M1082706.

[15] Stanislav Míka and Petr Vaněk. Acceleration of convergence of a two-level algebraic algorithm by aggregation in smoothing process. *Applications of Mathematics*, 37(5):343–356, 1992.

[16] Artem Napov and Yvan Notay. An algebraic multigrid method with guaranteed convergence rate. *SIAM Journal on Scientific Computing*, 34(2):A1079–A1109, 2012. doi: 10.1137/100818509.

[17] Yvan Notay. An aggregation-based algebraic multigrid method. *Electronic transactions on numerical analysis*, 37(6):123–146, 2010.

[18] Luke N. Olson, Jacob Schroder, and Raymond S. Tuminaro. A new perspective on strength measures in algebraic multigrid. *Numerical Linear Algebra with Applications*, 17(4):713–733, 2010.

[19] Florian Rathgeber, David A. Ham, Lawrence Mitchell, Michael Lange, Fabio Luporini, Andrew T. T. McRae, Gheorghe-Teodor Bercea, Graham R. Markall, and Paul H. J. Kelly. Firedrake: automating the finite element method by composing abstractions. *ACM Trans. Math. Softw.*, 43(3):24:1–24:27, 2016. ISSN 0098-3500. doi: 10.1145/2998441.

[20] J. W. Ruge and K. Stüben. Algebraic multigrid (AMG). In S. F. McCormick, editor, *Multigrid Methods*, volume 3 of *Frontiers in Applied Mathematics*, pages 73–130. SIAM, Philadelphia, PA, 1987.

[21] Nico Schlömer. pygmsh: A Python frontend for Gmsh. URL `https://github.com/nschloe/pygmsh`.

[22] K. Stüben. An introduction to algebraic multigrid. In U. Trottenberg, C. Oosterlee, and A. Schüller, editors, *Multigrid*, pages 413–528. Academic Press, London, 2001.

[23] U. Trottenberg, C. W. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, London, 2001.

[24] R. S. Tuminaro and C. Tong. Parallel smoothed aggregation multigrid: Aggregation strategies on massively parallel machines. In J. Donnelley, editor, *SuperComputing 2000 Proceedings*, 2000.

[25] P Vaněk, J Mandel, and M Brezina. Algebraic multigrid based on smoothed aggregation for second and fourth order problems. *Computing*, 56:179–196, 1996.

[26] Petr Vaněk. Acceleration of convergence of a two-level algorithm by smoothing transfer operators. *Applications of Mathematics*, 37(4):265–274, 1992.

Appendix A: Review of algebraic multigrid methods

Algebraic multigrid methods seek to approximate solutions to sparse linear systems of the form

$$Au = f \tag{5.12}$$

for $A \in \mathbb{R}^{N_{\text{node}} \times N_{\text{node}}}$, and $u, f \in \mathbb{R}^{N_{\text{node}}}$. Here, we outline *aggregation*-based AMG methods for use as an application in the development of Lloyd-style clustering. The set of indices, $\{1, \ldots, N_{\text{node}}\}$, enumerate the degrees of freedom (DoFs) and represent the fine level in the multilevel grid hierarchy. This set is partitioned and grouped into disjoint clusters, see Definition 5.1.

Each cluster represents a node in the coarse grid and, collectively, the cluster mapping defines a tentative restriction operator, $\hat{R}$, as

$$\hat{R}_{a,i} = \begin{cases} 1 & \text{if vertex } i \text{ is in cluster } a, \\ 0 & \text{otherwise.} \end{cases} \tag{5.13}$$

An example with 12 fine nodes and 3 coarse nodes (clusters) is given in Figure 5.12; the pattern for (the transpose of) $\hat{R}$ is also illustrated.

The restriction pattern defines the tentative interpolation pattern through $\hat{Z} = \hat{R}^T$. Smoothed aggregation (SA) AMG proceeds by using the nonzero pattern of $\hat{Z}$ as a partition of unity to localize a given global set of vectors, C, defining the near-null space of matrix A and, then, smoothing each column of the resulting matrix, Z, with (for example) weighted Jacobi. This defines the smoothed interpolation operator, Z, from which a coarse-level operator is defined over cluster DoFs as $A_c = Z^T A Z$.

The complete algorithm for constructing SA AMG is given in Algorithm 5.16, where we note the omission of several details (and optional parameters, denoted by [*opt*]) since the focus of this work is primarily on Line 4. We refer the reader to [6, 22, 25] for a more complete description and analysis of aggregation-based AMG methods. Here, we note that Line 3 is critically important to the convergence of the method; in practice, unit weights or algebraic distances *can* be used, yet generalized measures such as the *evolution* measure [18] (used in Section 5.4) have proven robust in practice.

With a multigrid hierarchy of coarse operators and interpolation, multigrid (MG) iterates via the familiar V-cycle as in Algorithm 5.17. In Section 5.4, we have considered both two-level ($N_{\text{level}} = 2$) and multilevel results, underscoring improved

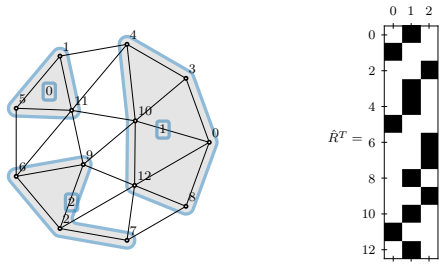


Figure 5.12: Example clustering and restriction matrix.

Algorithm 5.16 Smoothed aggregation — setup

function SA-SETUP(A_0, N_{level}, C)
  for $\ell \leftarrow 0, \ldots, N_{\text{level}} - 1$ **do**
    $W \leftarrow$ EDGE-WEIGHTS(A_ℓ, [*opt*]) ▷ determine strong edges in graph of A
    $m, c \leftarrow$ CLUSTER(W, [*opt*]) ▷ cluster membership and centers
    $Z_\ell \leftarrow$ INTERPOLATION(m, C, [*opt*]) ▷ form interpolation
    $A_{\ell+1} = Z_\ell^T A_\ell Z_\ell$ ▷ construct coarse-level operator
  return $\{A_\ell\}_0^{N_{\text{level}}}$, $\{Z_\ell\}_0^{N_{\text{level}}-1}$

convergence by improving the clustering, while leaving the other multigrid parameters untouched. As we use a subscript within these algorithms to denote the level within the multigrid hierarchy, we use a superscript to indicate the multigrid iteration number, with $u^{(k+1)} =$MG-V-CYCLE($A_0, \ldots, A_{N_{\text{level}}}$,$Z_0, \ldots, Z_{N_{\text{level}}-1}$, $u^{(k)}$, f).

Algorithm 5.17 MG cycle

function MG-V-CYCLE($A_0, \ldots, A_{N_{\text{level}}}$,$Z_0, \ldots, Z_{N_{\text{level}}-1}$, u_0, f_0)
  for $\ell = 0, \ldots, N_{\text{level}} - 1$ **do**
    $u_\ell \leftarrow$ RELAX(A_ℓ, u_ℓ, f_ℓ) ▷ fixed number of relaxation sweeps
    $f_{\ell+1} \leftarrow Z_\ell^T (f_\ell - A_\ell u_\ell)$ ▷ compute restricted residual
  $u_{N_{\text{level}}} \leftarrow A_{N_{\text{level}}}^{-1} f_{N_{\text{level}}}$ ▷ solve coarsest level problem
  for $\ell = N_{\text{level}} - 1, \ldots, 0$ **do**
    $u_\ell \leftarrow u_\ell + Z_\ell u_\ell$ ▷ interpolate and correct
    $u_\ell \leftarrow$ RELAX(A_ℓ, u_ℓ, f_ℓ) ▷ fixed number of relaxation sweeps
  return u_0

Appendix B: Notation

Table 5.2 summarizes the notation.

Symbol	Definition	Domain
A	left hand side operator in the linear system $Au = f$	$\mathbb{R}^{N_{\text{node}} \times N_{\text{node}}}$
a	cluster index	$\{1, \ldots, N_{\text{cluster}}\}$
B	set of border nodes between clusters; $B \subseteq V$	$\mathcal{P}(V)$
c_a	center node index for cluster a	V
c_a^1, c_a^2	new cluster centers if cluster a is split; see Algorithm 5.14	V
δ	second-term energy scaling coefficient, see (5.3)	$\mathbb{R}$
d_i	shortest-path distance to node i from nearest center; $d_i = \infty$ if node i is not in a cluster	$\overline{\mathbb{R}}_{\geq 0}$
$D_{i,j}$	shortest-path distance from node i to j within a single cluster; $D_{i,j} = \infty$ if there is no such path $i \to j$	$\overline{\mathbb{R}}_{\geq 0}$
$\Delta_{\min}$	minimum difference between distinct values of $W_{i,j}$	$\mathbb{R}$
E	set of edges in the graph G	$\mathcal{P}(V \times V)$
f	right hand side of the linear system $Au = f$	$\mathbb{R}^{N_{\text{node}}}$
G	graph with nodes V, edges E, and weights W	
H	shortest-path energy function, see (5.1)	$\mathbb{R}$
H_δ	energy function minimized by clustering, see (5.2)	$\mathbb{R}$
i, j, k	node indices	V
L_a	energy increase if cluster a is eliminated; see Algorithm 5.13	$\mathbb{R}$
M_a	whether cluster a is modifiable during rebalancing	$\{\text{True}, \text{False}\}$
m_i	cluster index (membership) containing node i	$\{1, \ldots, N_{\text{cluster}}\}$
N_{node}	number of nodes	$\mathbb{N}_1$
N_{cluster}	number of clusters	$\mathbb{N}_1$
n_i	number of nodes with i as predecessor	$\mathbb{N}_0$
$P_{i,j}$	predecessor index for node j on the shortest path $i \to j$ within a cluster; $P_{i,j} = 0$ if there is no path $i \to j$	$V \cup \{0\}$
p_i	predecessor index for node i on the shortest path from its cluster center; $p_i = 0$ if node i is not in a cluster	$V \cup \{0\}$
q_i	sum of squared distances from node i to all other nodes in the same cluster	$\mathbb{R}_{\geq 0}$
S_a	energy decrease if cluster a is split; see Algorithm 5.14	$\mathbb{R}$
s_a	size (number of nodes) of cluster a	$\mathbb{N}_1$
$\mathfrak{s}_i$	size (number of nodes) of the cluster containing node i; $\mathfrak{s}_i = 0$ if node i is not in a cluster	$\mathbb{N}_0$
T	total number of time/iteration steps taken by an algorithm ($T_{\max}$ and $T_{\text{BF}\max}$ denote the maximum)	$\mathbb{N}_0$
t	time/iteration index	$\mathbb{N}_0$
u	solution vector in the linear system $Au = f$	$\mathbb{R}^{N_{\text{node}}}$
V	set of nodes in the graph; $V = \{1, \ldots, N_{\text{node}}\}$	$\mathcal{P}(\mathbb{N}_1)$
V_a	set of nodes in cluster a; $V_a \subseteq V$	$\mathcal{P}(V)$
$W_{i,j}$	weighted adjacency matrix of the graph where $W_{i,j}$ is the edge weight $i \to j$	$\mathbb{R}$
z	generic variable placeholder	—

Table 5.2: List of symbols. Here $\mathcal{P}()$ denotes the power set and $\overline{\mathbb{R}}$ is the extended reals.

Chapter 6

Conclusions and Future Work

Few AMG methods offer convergence guarantees. Reduction-based AMG, AMGr, is one of them. A key element of the AMGr methodology is the partitioning of the DoFs into coarse and fine nodes. In this thesis, we develop a coarsening algorithm based on simulated annealing. The achievement of this part of the thesis is to show the successful application of randomized search and other derivative-free optimization algorithms to combinatorial optimization problems in AMG coarsening. Also the new algorithm improves the coarsening over the existing greedy algorithm. Subsequent work, by Taghibakhshi et al. [1], successfully apply machine learning tools to solve the same problem.

Classical AMGr yields poor convergence factors for the problems that are not diagonally dominant. In the second part of this thesis, we develop a new method using AMG heuristics to generalize classical AMGr. We apply the new method to different problems where the coefficient matrices are not diagonally dominant. The new method works comparatively well for a number of problems where classical AMGr works poorly.

Aggregation plays an important role in many different fields including multigrid methods for solving sparse linear systems. In the third part of the thesis, we introduce an algorithm that forms balanced aggregates over the graph of sparse matrix. The algorithm provides uniform, more rounded and better centered partitions of the graph. We establish linear complexity of the algorithm and also provide numerical results to show better partitioning of the nodes into aggregates and its effect on multigrid convergence.

Each part of the thesis elicits some questions to be studied and ideas for further studies. A few future research directions are:

- The main drawback of the coarsening algorithm presented in Chapter 3 is the high computational cost of simulated annealing. Finding a more efficient manner to approximate the functional would substantially reduce the computational cost. The research work shows that this combinatorial optimization problem can be solved using randomized search and other derivative-free optimization algorithms. Hence, the obvious next study in this direction would be to try other derivative-free optimization methodologies or machine learning strategies to develop coarsening algorithms with improved speed.

- Chapter 4 introduces a new class of AMGr methods. However, these methods are not covered by existing theoretical results for AMGr. Development of theory for the new class of methods would be an exciting research direction. Such theory could help us understand how the SPAI method helps to improve convergence and if this approach can be improved further. Some questions arise. How do we quantify approximations made by SPAI in a way that is productive for theory? Is it possible to generalize the existing AMGr theory to the new class of methods?

- Another research direction would be to try the new algorithm on a broader class of problems, including the Helmholtz equation, highly advective flows, etc. An important next step would be to make the method work for systems of PDEs.

- In our study of generalizing Lloyd's algorithm for graph clustering, we defined an energy functional and developed an algorithm to minimize the functional. However, it is quite possible that minimizing or maximizing a different energy functional would lead to a better aggregation method. Hence, developing new functionals to improve the aggregation method might be a good idea for a future research. Also, it is worth studying the effects of initial seeding on the overall computational cost of the aggregation to see if a good initial seeding helps to get better aggregates faster. It may also be worthwhile to employ machine learning techniques for the aggregation.

Bibliography

[1] Ali Taghibakhshi, Scott MacLachlan, Luke Olson, and Matthew West. Optimization-based algebraic multigrid coarsening using reinforcement learning.

In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 12129–12140. Curran Associates, Inc., 2021.