© 2007 by David Michael Alber. All rights reserved.

EFFICIENT SETUP ALGORITHMS FOR PARALLEL ALGEBRAIC MULTIGRID

BY

DAVID MICHAEL ALBER

B.S., University of Iowa, 1999
M.S., University of Illinois at Urbana-Champaign, 2004

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2007

Urbana, Illinois

Abstract

Solving partial differential equations (PDEs) using analytical techniques is intractable for all but the simplest problems. Many computational approaches to approximate solutions to PDEs yield large systems of linear equations. Algorithms known as linear solvers then compute an approximate solution to the linear system.

Multigrid methods are one class of linear solver and find an approximate solution to a linear system through two complementary processes: relaxation and coarse-grid correction. Relaxation cheaply annihilates portions of error from the approximate solution, while coarse-grid correction constructs a lower dimensional problem to remove error remaining after relaxation.

In algebraic multigrid (AMG), the lower dimensional space is constructed by coarse-grid selection algorithms. In this thesis, an introduction and study of independent set-based parallel coarse-grid selection algorithms is presented in detail, following a review of algebraic multigrid. The behavior of the Cleary-Luby-Jones-Plassmann (CLJP) algorithm is analyzed and modifications to the initialization phase of CLJP are recommended, resulting in the CLJP in Color (CLJP-c) algorithm, which achieves large performance gains over CLJP for problems on uniform grids. CLJP-c is then extended to the Parallel Modified Independent Set (PMIS) coarse-grid selection algorithm producing the PMIS-c1 and PMIS-c2 algorithms. Experimental results are provided for six problems run with a large collection of independent set-based coarsening algorithms.

The experimental results motivate the design of new coarsening algorithms to improve the performance of coarse-grid selection itself. A new algorithm labeled Bucket Sorted Independent Sets (BSIS) is developed and contributes two major advances. First, the cost of selecting independent sets while coarsening is substantially less expensive, with experiments demonstrating 23% savings over CLJP-c. Second, theory is developed proving that all generalized forms of the coarsening algorithms studied in this thesis using the same selection and update parameters choose identical coarse grids, given the same initial weights. The theory is powerful because it provides insight and enables the development of more efficient algorithms without affecting convergence properties.

Dedicated to my family and friends.

Acknowledgments

This dissertation is the result of nearly seven years of research while a member of the Scientific Computing Group in the Department of Computer Science at the University of Illinois at Urbana-Champaign (UIUC). I have been influenced by many people during my graduate career and am particularly grateful for the guidance and friendship of my advisors, Luke Olson and Paul Saylor. Since his arrival at UIUC in the fall of 2005, Luke has provided insight, direction, and focus to both my research and to the process of earning a doctorate. I appreciate his good humor and the support he has provided for the last several semesters. I am also grateful for years of support from Paul and credit him with starting me on the path that led to my research topic.

The other members of my dissertation committee have shared their time and insight with me repeatedly throughout the years. I first learned about parallel coarse-grid selection algorithms at a talk delivered by Rob Falgout at Lawrence Livermore National Laboratory (LLNL) on August 25, 2004. I was immediately interested and eventually began working on coarsening algorithms. Rob has always been helpful when I approached him with questions and offered outstanding suggestions for improving my thesis. I am appreciative for the many times Mike Heath met with me to discuss various issues. Among other benefits of his guidance, I discovered a broad personal interest in combinatorial scientific computing while reading a paper he recommended.

I am grateful for the opportunity to have spent three summers at LLNL. Jim Jones is an excellent supervisor and provided my first exposure to multigrid. I am glad to have worked with him on many occasions. I also enjoyed working with Barry Lee and appreciate the encouragement he provided. I have benefited repeatedly from the hard work of Linda Becker and Pam Mears to make the summer student program run efficiently. Bill Daniels was always friendly and the source of interesting conversations. Finally, I appreciate the long-time support from the Center for Applied Scientific Computing (CASC) at LLNL. In addition to working with CASC scientists, CASC has provided access to LLNL parallel machines.

Several other organizations have supported me and have my thanks. The Department of Energy has funded several years of research assistantships and has awarded me conference travel grants. SIAM funded a research assistantship for one semester and awarded me multiple conference travel grants. Thanks to NCSA for generously providing an Access Grid room for my preliminary and final exams.

The Scientific Computing Group at UIUC has been a source of collaboration and friendship. Prior to his departure to Virginia Tech, I had the opportunity to work with Eric de Sturler. Through the years, I have benefited from his teaching, insight, and collaboration. I have had the pleasure of knowing many graduate students in the Group during my tenure and thank them for enthralling conversations and good humor. I wish you all success. I owe particular thanks to Zhen Cheng, Bill Cochran, Eric Cyr, Rebecca Hartman-Baker, Vanessa Lopez, Hanna Neradt, Mike Parks, Chris Siefert, Ryan Szypowski, and Shun Wang for being good friends and colleagues and for helping me at various times during my graduate education. I appreciate the enormous help Trisha Benson has been in repeatedly navigating the treacherous waters of bureaucracy prior to and following travel and other events.

To my friends, I offer my greatest thanks. In addition to those named above, Kiran Lakkaraju, Ying Lu, Subhangan Venkatesan, Chen Wan, and many others have offered support and unique perspectives. We have been through a lot together, and I am fortunate to have met each of you. And to Chao-Jen Wong: meeting you one year ago was an unlikely event, but happened nonetheless. Thank you for encouragement, motivation, your sense of humor, and for making the future look much more interesting.

Finally, I am indebted to my family for years of support, and I am privileged to have extraordinarily supportive and loving parents. The devotion and guidance of Michael and Carol Alber are the source of my achievements. Thank you for being with me through this experience and all others before it.

Table of Contents

List of Abbreviations . . . . **x**

List of Algorithms . . . . **xi**

Chapter 1 Introduction **1**
1.1 Overview of Ideas 3
1.2 Organization 4

Chapter 2 Algebraic Multigrid **6**
2.1 Basic Concepts 6
2.1.1 Relaxation 7
2.1.2 Coarse-Grid Correction 8
2.2 Smooth Error 9
2.3 AMG Phases 10
2.3.1 Setup Phase 10
2.3.2 Solve Phase 14

Chapter 3 Coarse-Grid Selection **16**
3.1 Strength Graph 16
3.2 Survey 17
3.3 Ruge-Stüben Coarsening 18
3.4 H1 Parallel Coarse-Grid Selection 21
3.4.1 RS3 21
3.4.2 Cleary-Luby-Jones-Plassmann 21
3.4.3 Falgout 24
3.5 H1′ Parallel Coarse-Grid Selection 26
3.5.1 Parallel Modified Independent Set 27
3.5.2 Hybrid Modified Independent Set 28
3.5.3 Comparisons 28

Chapter 4 Color-Based Parallel Coarse-Grid Selection **30**
4.1 Analysis of CLJP 30
4.2 Modifying CLJP 33
4.2.1 Observations 33
4.2.2 Modifications 33
4.2.3 Parallel CLJP-c 36
4.2.4 Implementation Specifics 36
4.3 PMIS-c1 and PMIS-c2 37
4.4 Method Fingerprints 37
4.5 Experiments 38
4.5.1 Methods and Measures 39
4.5.2 Fixed Problem Sizes 42

4.5.3 Scaled Problem Sizes . . . 47
4.6 Conclusions . . . 62

Chapter 5 Bucket Sorted Independent Sets . . . 65
5.1 Introduction . . . 65
5.2 CLJP-c . . . 65
5.3 Coarse-Grid Selection Search and Weight Update . . . 66
5.4 Bucket Sorted Independent Set Selection . . . 67
5.4.1 Coarse Grid Invariance . . . 67
5.4.2 Bucket Sorted Independent Sets Algorithm . . . 71
5.5 Weight Update Aggregation . . . 74
5.6 Experimental Results . . . 74
5.7 BSIS Variants . . . 77
5.8 Parallelizing BSIS . . . 77

Chapter 6 Parallel Compatible Relaxation . . . 81
6.1 Intuition and Design . . . 81
6.2 Experimental Results . . . 82

Chapter 7 Conclusions . . . 88
7.1 Contributions . . . 88
7.2 Future Work . . . 90
7.3 Closing Remarks . . . 92

Appendix A Geometric Multigrid and the Two-Grid Operator . . . 93
A.1 Stencil Notation . . . 93
A.2 Basic Concepts . . . 94
A.3 Components of Multigrid . . . 94
A.3.1 Smoothers . . . 94
A.3.2 Coarsening Strategy . . . 95
A.3.3 Restriction . . . 95
A.3.4 Prolongation . . . 96
A.3.5 Coarse-Grid Operator . . . 96
A.4 Assembling the Two-Grid Operator . . . 97
A.4.1 Presmoothing . . . 97
A.4.2 Coarse-Grid Correction . . . 97
A.4.3 Postsmoothing . . . 98
A.4.4 Two-Grid Operator . . . 98

Appendix B Local Fourier Mode Analysis . . . 100
B.1 Basic Idea . . . 100
B.2 Smoothing Analysis . . . 102
B.3 Two-Grid Analysis . . . 103
B.4 Using Other Smoothers . . . 106
B.5 Computational LMA . . . 106

Appendix C Additional Experimental Results . . . 107
C.1 Fixed-Size 3D 7-Point Laplacian . . . 107
C.2 Fixed-Size 3D Unstructured Laplacian . . . 116
C.3 Scaled 3D 7-Point Laplacian . . . 124
C.4 Scaled 3D Unstructured Laplacian . . . 132
C.5 3D Unstructured Anisotropic Problem . . . 140
C.6 3D Laplacian Holes . . . 148

References . . . 156

Author's Biography **160**

List of Abbreviations

AMG	Algebraic Multigrid
BSIS	Bucket Sorted Independent Sets
CLJP	Cleary-Luby-Jones-Plassmann
CLJP-c	Cleary-Luby-Jones-Plassmann in Color
CR	Compatible Relaxation
HMIS	Hybrid Modified Independent Set
PMIS	Parallel Modified Independent Set
PMIS-c1	Parallel Modified Independent Set in Distance-One Color
PMIS-c2	Parallel Modified Independent Set in Distance-Two Color
RS	Ruge-Stüben

List of Algorithms

2.1 AMG Setup Phase 14
2.2 AMG Solve Phase 15
3.1 Ruge-Stüben Coarse-Grid Selection 19
3.2 Cleary-Luby-Jones-Plassmann (CLJP) 23
3.3 CLJP Independent Set Selection 23
3.4 CLJP Weight Update 24
3.5 Falgout Coarse-Grid Selection 25
3.6 Parallel Modified Independent Set (PMIS) 27
3.7 Hybrid Modified Independent Set (HMIS) 29
4.1 Weight Initialization for CLJP-c 35
5.1 Coarse-Grid Selection Graph Search 67
5.2 CLJP Weight Update for CSR Matrix 68
5.3 BSIS Data Structure Setup 71
5.4 Independent Set Selection 71
5.5 BSIS Weight Update 71
6.1 Compatible Relaxation 82

Chapter 1

Introduction

Many phenomena in science and engineering are modeled mathematically with partial differential equations (PDEs). Solving a PDE analytically is often intractable, and alternative methods, such as numerical approximation, are needed. A *discretization method*, such as finite differences or finite elements, is used to approximate the original problem at unknowns, typically on a mesh. This new problem takes the form of a linear system

$$Ax = b, \tag{1.1}$$

where A is an $n \times n$ matrix.

For finite differences and finite element methods using local support basis functions, A is *sparse*. That is, A contains few nonzeros, and if the mesh is refined and the problem rediscretized, the number of unknowns increases while the number of nonzeros per row in A remains nearly constant. If, on the other hand, finite elements with global support basis functions are used, A is usually *dense*.

Two broad groups of linear solution methods exist for solving (1.1): *direct* and *iterative*. Direct methods include Gaussian Elimination methods such as LU factorization, Cholesky factorization, multifrontal methods, and others. Direct methods are attractive when there is more than one right-hand side, b, because the factorization need only be computed once and is reusable. Assuming the system is nonsingular (and symmetric positive definite in the case of Cholesky) direct methods have the advantage that solution time is insensitive to the conditioning of the matrix. These methods, however, typically scale more poorly than iterative methods and are difficult to parallelize.

Unlike direct methods, iterative methods produce a sequence of approximate solutions:

$$x^{k+1} = x^k + \alpha^k d^k, \tag{1.2}$$

where x^k is the approximate solution in the kth iteration and $\alpha^k d^k$ is an update vector that removes components of the error from x^k. Sensible iterative methods use information about the system matrix when computing $\alpha^k d^k$ and yield a small cost per iteration compared to the cost of factoring the matrix with a direct method.

A popular class of iterative methods is the *Krylov subspace methods*, including the conjugate gradient (CG) method [38] and the generalized minimum residual (GMRES) method [50]. Krylov methods construct approximations from an increasingly larger space associated with A (i.e, the Krylov subspace). In each iteration k, $\alpha^k d^k$ is the vector in the Krylov subspace that minimizes the error or residual of the next approximate solution with respect to some norm.

Krylov methods are sensitive to the condition number of A, and *preconditioning* is often necessary for an effective iterative process. Preconditioners come in many forms, such as approximate factorizations (e.g., ILU [49]), approximate inverse preconditioners (e.g., SPAI [5]), and other iterative methods (e.g., Jacobi, SOR).

Another class of iterative solvers and preconditioning techniques is multigrid methods [14, 55]. Multigrid creates a hierarchy of linear systems, whereby the original problem is on the *fine grid*, and other levels in the hierarchy are *coarse grids* smaller in size. The method removes part of the error on each level of the hierarchy and relies on coarser levels to annihilate remaining error.

Within multigrid methods two broad classes exist: *geometric multigrid* and *algebraic multigrid (AMG)*. Geometric multigrid is restricted to problems on structured grids for which coarse levels are implicitly defined. Multigrid solvers are extended to problems on unstructured meshes by algebraic multigrid, which defines coarse grids explicitly using coarse-grid selection algorithms. AMG algebraically constructs the operators used to transfer information between levels in the grid hierarchy. These components are manufactured in the AMG *setup phase* and have a large impact on the effectiveness of the AMG *solve phase*.

An appealing feature of multigrid methods is that they converge in $O(n)$ time (i.e., independently of problem size) for a number of problems. This potential for optimality mixed with the potential for parallel implementation makes AMG an attractive algorithm to solve large-scale problems. Realizing reasonable scalability, however, implies both the solve phase and the setup phase must run efficiently.

Iterative methods parallelize more naturally than direct solvers, making them well-suited for solving very large, sparse linear systems. Using distributed memory multiprocessors as a computational platform presents a number of challenges, and the need to overcome problems of scale have continued to increase dramatically during the period in which this research was conducted. Table 1.1 lists several powerful machines when the author began and finished graduate studies.

Name (Location)	Rank (Date)	Nodes	Procs (per Node)	Memory	Peak
ASCI White (LLNL)	1 (11/2000)	512	8,192 (16)	16GB/node	12 Tflops
Red Storm (Sandia)	2 (11/2006)	12,960	25,920 (2)	3GB/node	124 Tflops
BlueGene/L (LLNL)	1 (11/2006)	65,536	131,072 (2)	512MB/node	367 Tflops

Table 1.1: Powerful computers at the beginning and end of the author's graduate education.

While computational power continues to increase (the first petaflops machine is expected in late 2008), the number of nodes and processors is also increasing. This trend increases the amount of communication in large parallel jobs and motivates the need for scalable parallel implementations.

Vertices on the fine grid in AMG typically depend only on nearby vertices and most reside on the same processor. On coarse grids, however, neighboring vertices tend to be further away. The trend continues on the coarsest grids where vertices that are separated by large distances on the fine level become neighbors. On parallel machines, this phenomenon is manifest by processors communicating with ever more "distant" processors. In a complete multigrid cycle, a processor has data dependencies with many more processors than it does on the fine level, so the changes seen in parallel architectures significantly impact AMG.

1.1 Overview of Ideas

This thesis focuses on and contributes to the study of parallel coarse-grid selection algorithms for AMG. These contributions follow fewer than eight years after the publication of the source of contemporary independent set-based parallel coarse-grid selection algorithms: the Cleary-Luby-Jones-Plassmann (CLJP) algorithm. In total, six new algorithms are developed in this thesis.

The algorithms this thesis develops improve the efficiency and effectiveness of their predecessors. Chapter 4 explores issues related to memory consumption and the convergence properties of AMG, which relates to the efficiency of the AMG solve phase. Following a study of performance of CLJP on structured grids, improvements are suggested and implemented. The resulting algorithm, CLJP-c, employs graph coloring algorithms to control coarse-grid structure and leads to significant improvements for structured problems. These ideas are applied to the Parallel Modified Independent Set (PMIS) coarsening algorithm to produce two additional algorithms.

Many coarse-grid selection algorithms utilize the concept of *strength of connection* to build coarse grids that accurately represent smooth error. In some situations, classical strength of connection does not provide accurate information. A different method called compatible relaxation produces coarse grids guaranteed to represent smooth error accurately. Two parallel compatible relaxation algorithms using concepts from earlier chapters are examined and implemented in Chapter 6.

This research and the research of others has produced a large and growing field of coarsening algorithms. These algorithms are introduced in several publications and are, in many cases, not tested against one another. Providing a single forum for all of the algorithms, this thesis contains a wealth of experiments and data examining the performance of many parallel independent set-based coarsening algorithms. This represents the largest set of coarsening algorithms tested simultaneously.

In Chapter 5, attention turns to the design and efficiency of coarsening algorithms themselves. Coarse-grid selection algorithms contain routines for searching a graph to identify new coarse-grid points. The weight initialization in CLJP and PMIS forces a brute force search, which involves large numbers of comparisons between vertex weights. The algorithms using graph coloring have a theoretical advantage in terms of the search methods available. An algorithm called Bucket Sorted Independent Sets (BSIS) is developed to use a bucket algorithm for sorting and identifying new coarse points without requiring comparisons between vertex weights. This novel application of comparison-free sorting produces a coarsening algorithm with much lower search costs. BSIS is the first coarse-grid selection algorithm developed with the explicit goal of reducing the cost of coarsening. In addition to presenting the new algorithm, theory is developed to prove that changes made from CLJP-c to BSIS do not affect the selected coarse grid.

1.2 Organization

The contents of this thesis are the union of several sources including three papers [2, 3], one which is not yet submitted for publication, and also new text. Material has been drawn from the author's Masters thesis [1] and resides primarily in Appendices A and B.

Algebraic multigrid and related concepts are introduced in Chapter 2. Chapter 3 focuses on coarse-grid selection. Strength of connection is revisited and the building blocks for independent set-based coarse-grid selection are outlined. Heuristics coarse-grid selection uses are introduced, followed by associated coarse-grid selection algorithms, such as the CLJP algorithm. In Chapter 4, the behavior of CLJP on structured grids is studied in depth, and the observations made are employed in the design of a new algorithm based on CLJP. This algorithm, CLJP-c, improves the performance of CLJP on structured grids, and a similar idea is applied to the PMIS algorithm to produce two additional algorithms: PMIS-c1 and PMIS-c2.

While the focus in Chapter 4 is on improving convergence properties of AMG through coarsening algorithm design, the attention in Chapter 5 is on the efficiency of coarse-grid selection itself. This study results in the BSIS algorithm. BSIS includes an independent set selection search that operates without comparing vertex weights, leading to significant decreases in the overall cost of coarse-grid selection.

Chapter 6 examines a different type of coarse-grid selection called compatible relaxation (CR). Parallel implementations are developed and the experiments presented demonstrate the promise of CR methods. Chapter 7 concludes the thesis by reiterating the major contributions of this research and includes a discussion for future directions in the study of parallel AMG setup phase algorithms.

Additional material is included in three appendices following Chapter 7. Appendix A contains an introduction of geometric multigrid. It is possible to predict the performance of geometric multigrid on various problems using local Fourier analysis. Appendix B introduces the Fourier analysis technique. The experiments in Chapters 3 through 6 produce abundant amounts of data. The corresponding plots visualize a limited selection of the data and do not provide information on other factors that are interesting to consider when studying coarse-grid selection. Additional data produced by the experiments is presented in tabular form in Appendix C.

Chapter 2

Algebraic Multigrid

Multigrid methods are versatile and applicable to a variety of problems, such as linear and nonlinear systems, optimization, and preconditioning, to name a few. The first multigrid methods were *geometric multigrid methods* [30, 4, 7]. Geometric multigrid depends on a predefined hierarchy of grids and operators to transfer information between levels. A grid is related to neighboring levels by containing a subset or superset of the grid points of its neighbors. The original problem is defined on the *fine grid*. All other grids in the hierarchy are *coarse grids*. A common type of coarsening for geometric multigrid is $h \rightarrow 2h$ coarsening and is illustrated in Figure 2.1.

Algebraic multigrid (AMG) [48, 9, 53, 14] was developed to provide flexibility to solve unstructured problems and anisotropic problems with multigrid. It generalizes the concepts of geometric multigrid and uses a coarse-grid selection algorithm, rather than depending on a predefined coarse-grid hierarchy.

AMG gained popularity due to its efficiency in solving certain classes of large, sparse linear systems and has been shown to converge independently of problem size for a number of problems [19, 3]. Due to this potential for optimality, AMG is an attractive algorithm for solving large-scale problems in parallel. However, to realize reasonable scalability in AMG, all components of the method must run well in parallel.

The terminology used for AMG is based on geometric multigrid terminology. Grid, grid point, smoother, and smooth error are commonly used terms used in AMG. Physical counterparts, however, do not exist. For example, algebraically smooth error in AMG is not necessarily geometrically smooth. Smoothness is identified through relaxation and the linear system.

2.1 Basic Concepts

Multigrid utilizes two complementary processes to iteratively eliminate error in an approximate solution to a linear system $Ax = b$: *relaxation* (or *smoothing*) and *coarse-grid correction.*

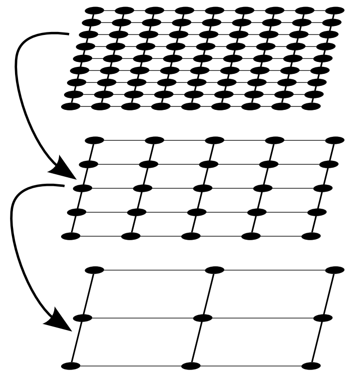


Figure 2.1: Geometric multigrid coarsening on the unit square. In $h \to 2h$ coarsening, the grid spacing is halved.

2.1.1 Relaxation

On each level in the hierarchy, multigrid applies an iterative method to quickly annihilate certain components of the error. This process is called relaxation. Many relaxation schemes are *stationary iterative methods*, which are applicable as solvers, but while they remove some error quickly, other components of the error are largely unaffected. Furthermore, for some methods, convergence is only guaranteed under restrictive conditions.

The most popular stationary iterative methods are based on a matrix splitting of the form $A = M - N$. The method is expressed as

$$x^{k+1} = M^{-1}Nx^k + M^{-1}b. \tag{2.1}$$

Equation (2.1) is often rewritten to include A:

$$x^{k+1} = (I - M^{-1}A)x^k + M^{-1}b. \tag{2.2}$$

By defining the smoothing operator $S = (I - M^{-1}A)$ and $c = M^{-1}b$, (2.1) simplifies to

$$x^{k+1} = Sx^k + c. \tag{2.3}$$

Multiple iterations are expressible as a stationary iterative method. For example, ν applications of the smoother is expressed as

$$x^{k+\nu} = S^{\nu}x^k + \tilde{c}, \tag{2.4}$$

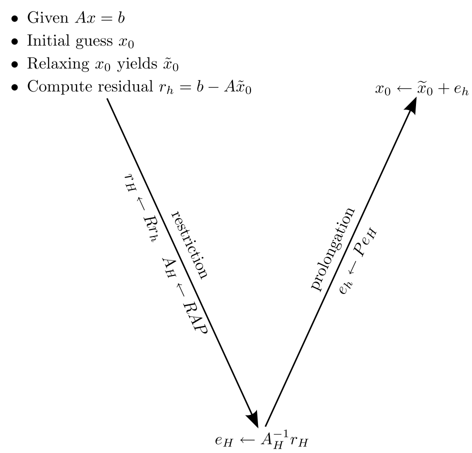


Figure 2.2: Two-grid cycle.

where $\tilde{c} = (S^{\nu-1} + S^{\nu-2} + \cdots + S + I)c$.

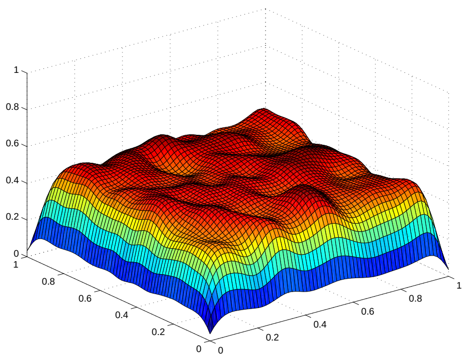
2.1.2 Coarse-Grid Correction

Assume $Ax = b$ has been smoothed, yielding the approximate solution $\hat{x}$, and define the *residual* as $r = b - A\hat{x}$. Also, define the *error*, $e = x^* - \hat{x}$, where x^* is the true solution. The error and the residual are related by the *defect equation*,

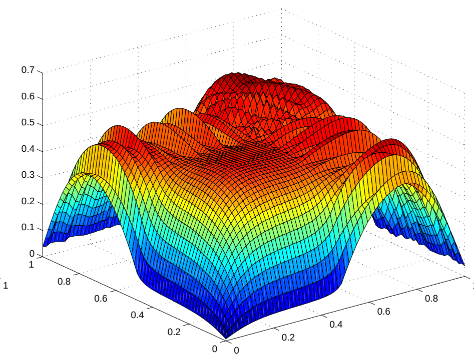
$$Ae = r. \tag{2.5}$$

These equations form the basis of *iterative refinement*. Solving (2.5) gives an update that leads to the exact solution since $x^* = \hat{x} + e$. The cost of solving (2.5), however, is equivalent to solving the original system. Multigrid seeks an approximation to e by transferring the residual onto the coarse grid through a method called *restriction*. The defect equation is solved on the coarse grid for e_H, and e_H is transferred back to the fine grid through *interpolation*. Finally, the solution is refined by coarse-grid correction: $x = \hat{x} + e$.

This process is known as the *two-grid cycle* and is illustrated in Figure 2.2. For clarity, fine-level vectors may retain a subscript h, while coarse level entities are denoted by H. The two-grid cycle is the most basic multigrid scheme because it utilizes a single coarse grid. Despite its simplicity, it contains the basic ideas of more complicated cycles.



(a) Geometrically smooth error



(b) Algebraically smooth error

Figure 2.3: Smooth error.

2.2 Smooth Error

When a relaxation scheme begins to stall, the error that remains is considered *smooth error*. Geometric multigrid requires this error to be locally smooth smooth (see Figure 2.3(a)). This property must be satisfied by selecting an appropriate relaxation scheme. In algebraic multigrid, smooth error does not necessarily satisfy the same conditions of geometric smoothness (see Figure 2.3(b)). The smooth error in both cases is, however, the result of the relaxation scheme stalling.

By examining relaxation further, insight is gained into the nature of smooth error. Starting from (2.1), error propagation through the method is expressed as:

$$
\begin{aligned}
x^* - x^{k+1} &= (I - M^{-1}A)x^* + M^{-1}b - \left((I - M^{-1}A)x^k + M^{-1}b\right), && (2.6)\\
e^{k+1} &= (I - M^{-1}A)(x^* - x^k), && (2.7)\\
e^{k+1} &= (I - M^{-1}A)e^k. && (2.8)
\end{aligned}
$$

Smooth error, therefore, satisfies $e \approx (I - M^{-1}A)e$. This expression is true when $M^{-1}Ae \approx 0$.

More specifically, for methods such as weighted Jacobi or Gauss-Seidel the smooth error expression is reduced, through additional analysis, to

$$Ae \approx 0, \tag{2.9}$$

implying that smooth error is composed of eigenvectors whose eigenvalues are close to zero (i.e. are near-nullspace). Additionally, it states that smooth error has a small residual.

2.3 AMG Phases

Algebraic multigrid algorithms execute in two phases: the *setup phase* and the *solve phase*. The AMG setup phase is responsible for selecting coarse grids, building prolongation and restriction operators, and constructing the coarse-level operators. The solve phase uses these products and implements a multigrid cycle using relaxation, restriction, and prolongation.

2.3.1 Setup Phase

In geometric multigrid, the coarse-grid hierarchy is fixed and available since the problem is structured. A natural grid hierarchy is not available to AMG since there is no assumption regarding structured grids. The purpose of the setup phase is to generate coarse grids and transfer operators, which are available directly to geometric multigrid for basic problems. It is noteworthy that geometric multigrid may also require an expensive setup phase for more complicated problems.

Coarse-Grid Selection

Coarse-grid selection is the processes of creating the degrees of freedom of a coarse-level problem. Classical forms of AMG use a subset of the fine-level unknowns as the coarse-level unknowns. This is called a *C/F splitting*, where C-points are variables that exist on both the fine and coarse levels and F-points are variables only on the fine level. This thesis studies algorithms used to select a C/F splitting in depth. Another form of algebraic multigrid known as *smoothed aggregation* [25, 13] forms aggregates of fine-level unknowns. These aggregates become coarse-level unknowns.

All AMG algorithms select coarse grids to accurately represent smooth error, to be suitable for accurately interpolating vectors from the coarse grid to the fine grid, and to be significantly smaller than the fine grid. It is easy to satisfy the first two properties by selecting a large coarse grid. Such a strategy, however, leads to an expensive coarse-grid problem.

These requirements are typically satisfied with a set of heuristics that use the *strength of connection* between coupled degrees of freedom. The classical strength of connection measure is based on the magnitude of off-diagonal entries in A. The set of unknowns that unknown i strongly depends upon is defined as

$$S_i = \left\{ j : i \neq j \text{ and } |a_{ij}| \geq \theta \max_{k \neq i} |a_{ik}| \right\}, \tag{2.10}$$

where a_{ij} is the entry in row i, column j of matrix A and $0 < \theta \leq 1$. Often, θ is 0.25. The set of unknowns that i strongly influences, denoted S_i^T, is defined as the set of unknowns that strongly depend on i:

$$S_i^T = \{j : i \in S_j\}. \tag{2.11}$$

The sets S_i and S_i^T directly correspond to the nonzero entries in the *strength matrix* S. Each nonzero in S corresponds to a strong connection between unknowns, where an entry in the strength matrix is defined as

$$S_{ij} = \begin{cases} 1 & \text{if } i \neq j \text{ and } |a_{ij}| \geq \theta \max_{k \neq i} |a_{ik}|, \\ 0 & \text{otherwise.} \end{cases} \tag{2.12}$$

This strength of connection measure is used in combination with two heuristics to define a valid coarse grid. These heuristics are as follows:

H1: For each unknown j that strongly influences F-point i, j is either a C-point or strongly depends on a C-point k that also strongly influences i.

H2: The set of C-points needs to form a maximal independent set in the reduced graph of S such that no C-point strongly depends on another C-point.

Note that heuristics H1 and H2 cannot generally be satisfied simultaneously. H1 is required by the classical AMG interpolation scheme, so it must be satisfied. H2, on the other hand, is used to guide the selection of coarse grids with few C-points.

Different heuristics exist and are used by other methods. See Chapter 3 for details of alternative heuristics and the algorithms that utilize them.

Transfer Operator Construction

Prolongation operators transfer vectors from coarse levels to finer levels: $Pe_H = e_h$. Construction of P is algebraically based on entries in A and the C/F splitting is computed by coarse-grid selection. During prolongator construction, the nonzero entries in prolongation operator matrix P are determined. These nonzero entries correspond to weights w in

$$Pe_i = \begin{cases} e_i & \text{if } i \in C, \\ \sum_{j \in C_i} w_{ij} e_j & \text{if } i \in F. \end{cases} \tag{2.13}$$

It was demonstrated in Section 2.2 that smooth error corresponds to a relatively small residual.

In terms of the residual of row i in A, this is expressed as

$$a_{ii}e_i + \sum_{j \in S_i} a_{ij}e_j \approx 0.$$

Therefore, smooth error at unknown i is approximated using the error at the strongly influencing neighbors of i:

$$a_{ii}e_i \approx -\sum_{j \in S_i} a_{ij}e_j. \tag{2.14}$$

Only C-points provide information in direct interpolation, so strongly connected F-points have no influence on the result of interpolation for i. Standard interpolation, on the other hand, includes information from strongly connected F-points by interpolating through mutual C-points. H1 ensures each pair of strongly connected F-points share at least one common strongly influencing C-point, yielding a well-defined interpolation procedure.

In standard interpolation, any j with $a_{ij} \neq 0$ is placed into one of three three groups: the set of strongly connected C-points, called the coarse interpolatory set, C_i, of i, the set of strongly connected F-points, D_i^s, and the set of weakly connected F- and C-points, D_i^w. Rewriting (2.14) in terms of these three sets yields

$$a_{ii}e_i \approx -\left(\sum_{j \in C_i} a_{ij}e_j + \sum_{j \in D_i^s} a_{ij}e_j + \sum_{j \in D_i^w} a_{ij}e_j\right). \tag{2.15}$$

To obtain an approximation for e_i, for $i \in F$, a new expression in terms of e_j, $j \in C_i$, is constructed. To do this, the contributions from D_i^s and D_i^w in (2.15) must be replaced with terms designed to approximate their values.

The unknowns in D_i^w are not strongly connected to i, so the error of any $j \in D_i^w$ has little influence on the error at i. Furthermore, (2.10) ensures the absolute value of a_{ij} for $j \in D_i^w$ is relatively small. In standard interpolation, the error from weakly connected neighbors is interpolated by replacing e_j for all $j \in D_i^w$ with e_i. This approximation changes (2.15) to

$$\left(\sum_{j \in D_i^w} a_{ij} + a_{ii}\right)e_i \approx -\left(\sum_{j \in C_i} a_{ij}e_j + \sum_{j \in D_i^s} a_{ij}e_j\right). \tag{2.16}$$

The requirement placed on strongly connected F-points is utilized in approximating the influence of all $j \in D_i^s$ using unknowns in C_i:

$$e_j \approx \frac{\sum_{k \in C_i} a_{jk} e_k}{\sum_{k \in C_i} a_{jk}}. \tag{2.17}$$

Notice that if H1 is not completely satisfied, (2.17) the denominator is not well-defined for some i and j. Following this approximation, (2.16) becomes

$$\left(\sum_{j \in D_i^w} a_{ij} + a_{ii} \right) e_i \approx - \left(\sum_{j \in C_i} a_{ij} e_j + \sum_{j \in D_i^s} a_{ij} \frac{\sum_{k \in C_i} a_{jk} e_k}{\sum_{k \in C_i} a_{jk}} \right). \tag{2.18}$$

This provides an approximation to e_i only in terms of error at unknowns in C_i (note the variable name change for D_i^s and D_i^w to m and n, respectively, for clarity):

$$e_i \approx - \left(\frac{\sum_{j \in C_i} a_{ij} e_j + \sum_{m \in D_i^s} a_{im} \frac{\sum_{k \in C_i} a_{mk} e_k}{\sum_{k \in C_i} a_{mk}}}{a_{ii} + \sum_{n \in D_i^w} a_{in}} \right). \tag{2.19}$$

Further manipulation yields

$$e_i \approx - \sum_{j \in C_i} \left(\frac{a_{ij} + \sum_{m \in D_i^s} \frac{a_{im} a_{mj}}{\sum_{k \in C_i} a_{mk}}}{a_{ii} + \sum_{n \in D_i^w} a_{in}} \right) e_j. \tag{2.20}$$

This result is in the form of (2.13) for $i \in F$. Therefore, the interpolation weight in standard interpolation is

$$w_{ij} = - \frac{a_{ij} + \sum_{m \in D_i^s} \frac{a_{im} a_{mj}}{\sum_{k \in C_i} a_{mk}}}{a_{ii} + \sum_{n \in D_i^w} a_{in}}. \tag{2.21}$$

Coarse-Level Operators

In algebraic multigrid, the coarse-level operator is typically the product RAP, where R and P are the restriction and prolongation operators, respectively, and $R = P^T$. This coarse-level operator is the *Galerkin operator*. Coarse-grid correction for the two-grid method using the Galerkin operator yields vector v in Range(P) minimizing $\|e_h - Pv\|_A$, where $\| \cdot \|_A$ denotes the A-norm.

While the Galerkin product is convenient for algebraic theory, unintended side-effects on the algorithmic complexity occur: the triple matrix product increases the density of the coarse-grid operator.

Setup Phase Deliverables

The AMG setup phase produces operators and sets of unknowns for each level in the multigrid hierarchy. Below, subscripts denote the grid level, where level zero is the finest level. Therefore, $A_0 = A$ and $\Omega_0 = \Omega$, where Ω is the index set relating to unknowns on a level. The sets of output produced are listed below.

Grids:	$\Omega_0 \supset \Omega_1 \supset \ldots \supset \Omega_M$
Grid operators:	$A = \{A_0, A_1, \ldots, A_M\}$
Prolongation operators:	$P = \{P_0, P_1, \ldots, P_{M-1}\}$
Restriction operators:	$R = \{R_0, R_1, \ldots, R_{M-1}\}$

The AMG setup phase is shown in Algorithm 2.1.

Algorithm 2.1 AMG Setup Phase

AMG-SETUP(A_1) {

$k \leftarrow 0$
while $|\Omega_k| >$ stopping size **do**
  $\Omega_{k+1} \leftarrow$ COARSE-GRID-SELECTION(A_k)
  $P_k \leftarrow$ BUILD-PROLONGATOR(A_k, Ω_{k+1}) /* *prolongation operator from level* $k+1$ *to* k */
  $R_k \leftarrow (P_k)^T$ /* *restriction operator from level* k *to* $k+1$ */
  $A_{k+1} \leftarrow R_k A_k P_k$ /* *Galerkin operator* */
  $k \leftarrow k+1$
end while
$m \leftarrow k$ /* *store number of levels in hierarchy* */

}

2.3.2 Solve Phase

The products of the setup phase are input to the solve phase, which implements relaxation and coarse-grid correction. The number of presmoothing and postsmoothing sweeps performed is defined by ν_1 and ν_2, respectively, and the order in which and the frequency of visits to a given level is defined by the type of *multigrid cycle* used.

Common multigrid cycles are the V-cycle and the W-cycle, as depicted in Figure 2.4. These cycles are defined recursively using the cycle index γ, which controls the number of times a coarse level descends to a coarser level before ascending to the next fine level. For example, in a V-cycle $\gamma = 1$, meaning each coarse level only restricts one time before interpolating a correction to the next finer level. Because increasing γ exponentially increases the number of times coarse levels are visited, only $\gamma = 1$ and $\gamma = 2$ are used in practice.

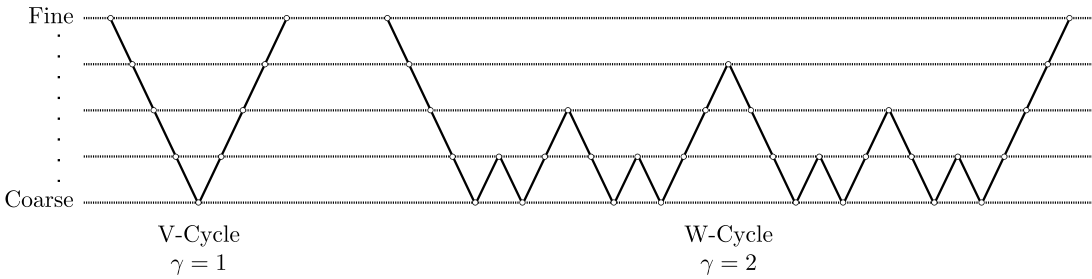


Figure 2.4: V-cycle and W-cycle.

Relaxation, the transfer operators, and the multigrid cycle are combined to form the AMG solve phase algorithm, of which a single iteration is outlined in Algorithm 2.2. The $k = 0$ in the function header is a default value to be used when no value is passed.

Algorithm 2.2 AMG Solve Phase

AMG($A,\ x,\ b,\ \nu_1,\ \nu_2,\ R,\ P,\ m,\ \gamma,\ k = 0$) {
if $k < m$ **then**
    $x \leftarrow \textsc{Smooth}(A_k, x, b, \nu_1)$ /* *presmooth: apply smoother ν_1 times* */
    $r \leftarrow b - A_k x$
    $r_c \leftarrow R_k r$ /* *restriction* */
    if $k \neq 0$ **then**
        for $i = 0$ to γ **do**
            $e_c \leftarrow \text{AMG}(A,\ \mathbf{0},\ r_c,\ \nu_1,\ \nu_2,\ R,\ P,\ m,\ \gamma,\ k+1)$
        end for
    else /* *on fine level* */
        $e_c \leftarrow \text{AMG}(A,\ \mathbf{0},\ r_c,\ \nu_1,\ \nu_2,\ R,\ P,\ m,\ \gamma,\ k+1)$
    end if
    $x \leftarrow x + P_k e_c$ /* *prolongation and update* */
    $x \leftarrow \textsc{Smooth}(A_k, x, b, \nu_2)$ /* *postsmooth: apply smoother ν_2 times* */
    return x
else /* ***on coarsest level*** */
    Solve $A_k x = b$
    return x
end if
}

Chapter 3

Coarse-Grid Selection

The focus of this thesis is on independent set-based methods, which form the largest class of coarse-grid selection algorithms. Given heuristic H2, it is sensible to employ independent set methods because they provide an approach to satisfying heuristic H1 while implicitly using H2 as a guideline. Figure 3.1 diagrams the relationships between many independent set-based coarse-grid selection methods. Five coarse-grid selection algorithms are introduced in this chapter: the classical Ruge-Stüben algorithm, Cleary-Luby-Jones-Plassmann (CLJP), Falgout, Parallel Modified Independent Set (PMIS), and Hybrid Modified Independent Set (HMIS). In Chapter 4, the behavior of CLJP on structured problems is studied in detail, and Chapters 4 and 5 continue the discussion with independent set-based algorithms that utilize graph coloring techniques.

3.1 Strength Graph

In the absence of a grid, AMG coarsening algorithms coarsen graphs. Coarse-grid selection generates a strength matrix to determine which unknowns influence each other strongly. Introduced in the previous chapter, classical strength of connection (2.12) is based on the following bound [14]:

$$\sum_{j \neq i} \left(\frac{|a_{ij}|}{a_{ii}} \right) \left(\frac{e_i - e_j}{e_i} \right)^2 \ll 1. \tag{3.1}$$

This bound is true for all i in a symmetric M-matrix. When $|a_{ij}|/a_{ii}$ is large (e.g., when i strongly depends on j), then e_i and e_j must be nearly equal. This observation enables the selection of coarse grids for which accurate interpolation is possible.

Equation (3.1) is not guaranteed to hold when A is not an M-matrix, resulting in coarse grids that ineffectively represent smooth error. New strength measures are an area of active research [11], and these measures may extend the applicability of AMG. This issue, however, does not directly impact coarsening design since the strength matrix is used as input rather than being generated

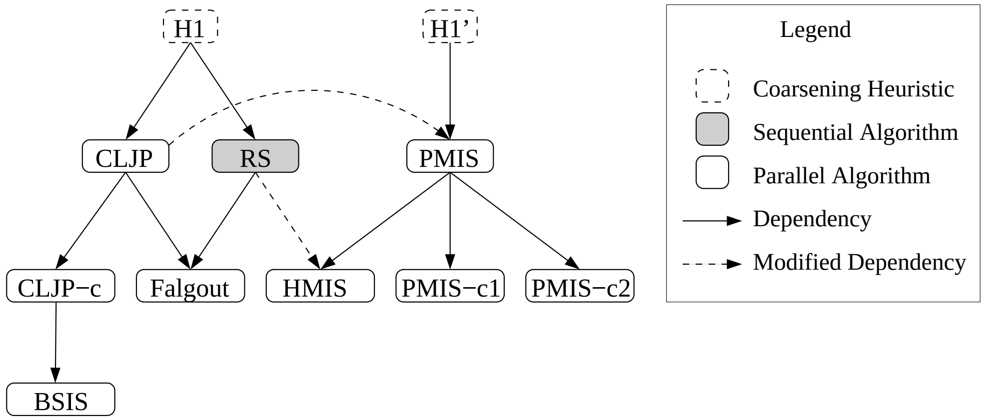


Figure 3.1: A partial taxonomy of independent set-based coarse-grid selection algorithms.

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \longleftrightarrow$$

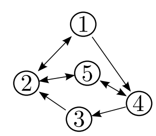


Figure 3.2: A strength matrix and associated graph. Edges point in the direction of dependence. For instance, an edge exists from Vertex 1 to Vertex 4 because Vertex 1 strongly depends on Vertex 4, as indicated by nonzero in the first row, fourth column of the matrix.

by the method itself. The guidelines used for coarse-grid selection are independent of the type of strength measure used. New strength measures are, therefore, applicable to coarse-grid selection without modifying the coarsening algorithms developed herein.

Coarse-grid selection works on a vertex-weighted graph of the strength matrix

$$G(S) = (V, E), \tag{3.2}$$

where V is the vertex set and E is the edge set. That is, the matrix S serves as the *adjacency matrix* for a graph from which coarse degrees of freedom are selected. Figure 3.2 depicts a matrix and its associated graph.

3.2 Survey

The coarse-grid selection algorithms of early AMG methods are based on the Ruge-Stüben (RS) [48] coarsening method, which is inherently sequential since only one C-point is selected in each iteration. When parallel AMG methods are considered, it is apparent the solve phase parallelizes in the same manner as geometric multigrid methods. The challenge is to parallelize the AMG setup phase. In particular, parallelization of the coarse-grid selection algorithm presents the largest obstacle [17].

Several methods have been proposed, the first of which are "parallel Ruge-Stüben" methods. The Cleary-Luby-Jones-Plassmann (CLJP) algorithm [18] followed and is a fully parallel algorithm which combines the two pass method of the Ruge-Stüben algorithm into a single parallel pass. Hybrid methods combining CLJP and Ruge-Stüben algorithms, have also been studied [37], notably Falgout coarsening. To lower the memory demands of coarse-grid hierarchies, Parallel Modified Independent Set (PMIS) and Hybrid Modified Independent Set (HMIS) were developed. These methods weaken the heuristics used for coarse-grid selection and require different interpolation operators.

Other methods take substantially different approaches to coarsening. The coarsening technique in [36] develops a parallel coarsening algorithm that utilizes RS. On each processor, RS selects several different coarse grids, and then one coarse grid is selected per processor in a way to minimize special treatment on processor boundaries. In [47], a greedy approach is used to produce a good splitting of the unknowns into fine-grid and coarse-grid sets. Subdomain blocking techniques [43] offer another approach for parallel coarse-grid selection by decoupling coarse grids and alleviating the need for communication on coarse levels. In another form of AMG called smoothed aggregation [25, 24, 13], coarse-level variables are selected by aggregating fine-level variables, rather than selecting subsets of fine-level variables.

3.3 Ruge-Stüben Coarsening

The classical coarse-grid selection algorithm is Ruge-Stüben (RS) coarsening, which is a two pass algorithm designed to satisfy H1. The first pass selects a maximal independent set guaranteeing that each F-point strongly depends on at least one C-point. It is possible after the first pass that the second condition in H1 remains unsatisfied. That is, there may be strongly connected F-points not sharing a common C-point neighbor. To remedy this, RS incorporates a second pass to locate and fix each of these instances by changing one of the two F-points to a C-point.

The methods studied in this thesis construct independent sets based on vertex weights in *neighborhoods*.

Definition 3.3.1. *The* neighborhood *of vertex i, denoted $\mathcal{N}_i$, is the union of the sets containing the strong influences of i and the strong dependencies of i: $\mathcal{N}_i = S_i \cup S_i^T$.*

The RS algorithm begins by initializing the weight of each vertex to the number of vertices it strongly influences (Figure 3.3(a)). Following initialization, RS begins the first pass. Each iteration of the first pass selects an unassigned vertex with the largest weight in the graph and makes it a C-point. The neighbors of this C-point become F-points, and the weights of unassigned neighbors of the newly selected F-points are incremented by one. The first pass is illustrated in parts (b)-(k) of Figure 3.3. The second pass of the algorithm traverses the graph searching for strong F-F connections with no common strongly influencing C-point. Two instances of strong F-F connections without a common strongly influencing C-point are shown in Figure 3.3(l), and the result of the second pass is depicted in Figure 3.3(m). RS coarsening is outlined in Algorithm 3.1.

Algorithm 3.1 Ruge-Stüben Coarse-Grid Selection

RUGE-STÜBEN(S, $C = \emptyset$, $F = \emptyset$) {
for all $i \in V$ **do** /* ***initialization*** */
    $w_i \leftarrow |S_i^T|$ /* *number of vertices strongly influenced by* i */
end for
while $C \cup F \neq V$ **do** /* ***first pass*** */
    Select $i \in V$ such that $w_i \geq w_j$ for all $j \in V$
    $C \leftarrow C \cup \{i\}$
    $F_{\text{new}} \leftarrow \{\text{unassigned vertices } j \in \mathcal{N}_i\}$
    $F \leftarrow F \cup F_{\text{new}}$
    for all $j \in F_{\text{new}}$ **do**
        for all $k \in \mathcal{N}_j$ **do**
            if k is unassigned vertex **then**
                $w_k \leftarrow w_k + 1$
            end if
        end for
    end for
end while
for all $i \in F$ **do** /* ***second pass*** */
    for all $j \in S_i \cap F$ **do**
        if $S_i \cap S_j \cap C = \emptyset$ **then**
            $F \leftarrow F \setminus \{j\}$
            $C \leftarrow C \cup \{j\}$
        end if
    end for
end for
return (C, F)
}

One goal of RS coarsening is to ensure that it produces the same result as standard coarsening for structured problems (Figure 2.1). A method used by RS to mimic standard coarsening is the incrementation of weights of vertices two hops from new C-points.

The RS algorithm selects only a single C-point per iteration, making it inherently sequential. New methods were needed to select coarse grids when the graph is distributed across several processors. The algorithms discussed in the remainder of this chapter and in Chapter 4 were developed in

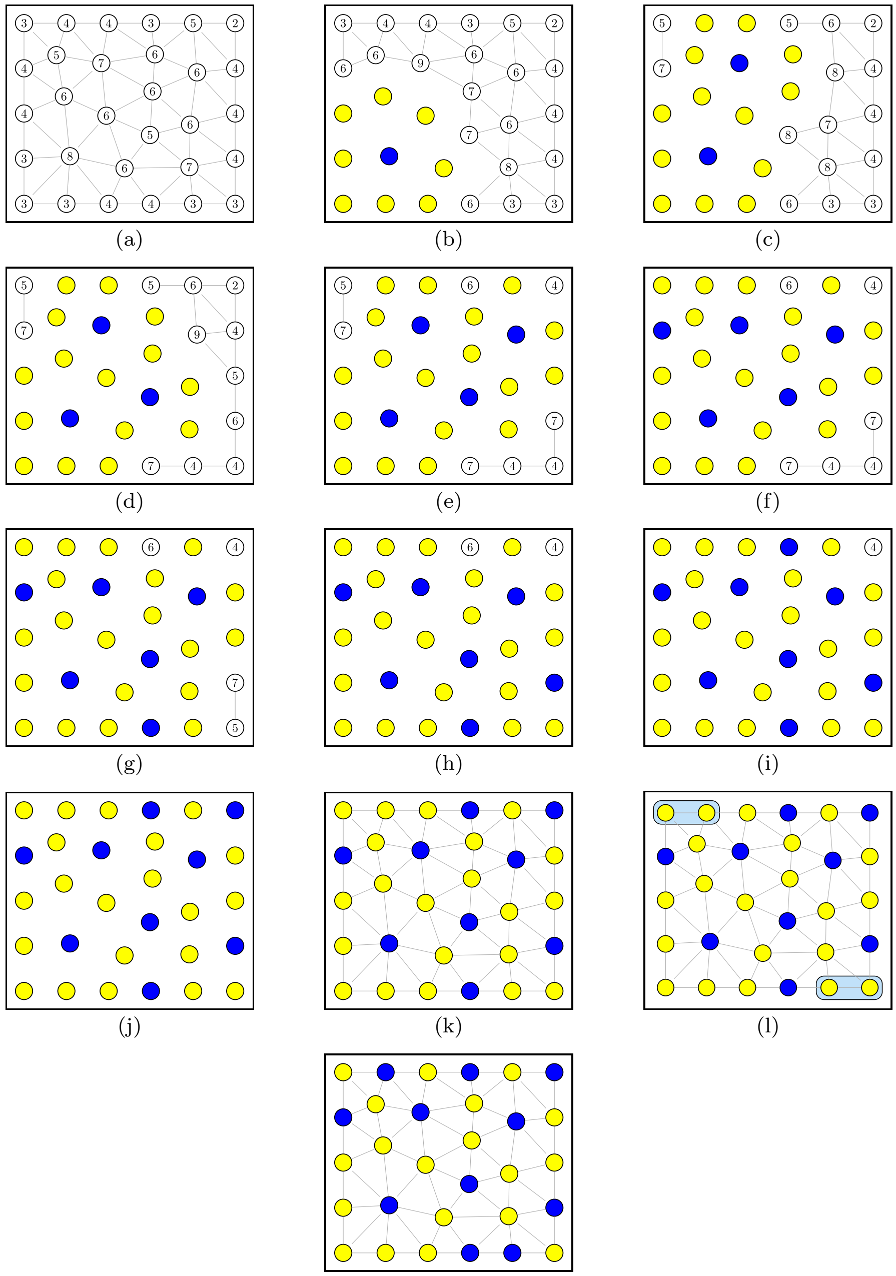


Figure 3.3: Ruge-Stüben coarse-grid selection. Unassigned vertices are white, F-points are yellow, and C-points are blue. Weight initialization is shown in (a). The first RS pass is illustrated in (b)-(k), and the second pass is in (l) and (m).

response to this need.

3.4 H1 Parallel Coarse-Grid Selection

Two classes of parallel independent set-based coarsening algorithms have been developed. Recall heuristic H1 from Section 2.3.1:

H1: For each unknown j that strongly influences F-point i, j is either a C-point or strongly depends on a C-point k that also strongly influences i.

3.4.1 RS3

The first parallel coarse-grid selection methods sought to parallelize RS, rather than creating new methods. A straightforward approach is to run RS independently on each processor domain. This leads to a valid coarsening on the interior of each processor, but along processor boundaries it is likely for H1 to be violated. By employing an additional pass along the processor boundaries after the usual two RS passes, RS3 is able to correct H1 violations by converting one of the F-points involved to a C-point. This approach often leads to a higher density of C-points on processor boundaries than in the interior, which increases communication and memory costs for the AMG method. Figure 3.4 demonstrates RS3 coarsening.

3.4.2 Cleary-Luby-Jones-Plassmann

The Cleary-Luby-Jones-Plassmann (CLJP) coarse-grid selection algorithm [18] was the first truly parallel coarsening algorithm. Through the use of a parallel independent set algorithm based on Luby's maximal independent set algorithm [45], CLJP selects many C-points in each iteration. Furthermore, CLJP is able to produce the same coarse grids regardless of how the data is partitioned across processors, given the same initial conditions. Algorithm 3.2 outlines CLJP coarsening.

The contribution of the random number in Line 2 is to create unique values for vertices that strongly influence the same number of neighbors. This augmentation generates many vertices in the graph whose weights are maximal within their neighborhood, which leads to CLJP's ability to select multiple C-points in each iteration. The selection of the independent set in Line 5 is done so that all vertices with the maximum weight in their neighborhood are placed in D (see Algorithm 3.3). That is,

$$D = \{i : w_i > w_j, \forall j \in \mathcal{N}_i\}. \tag{3.3}$$

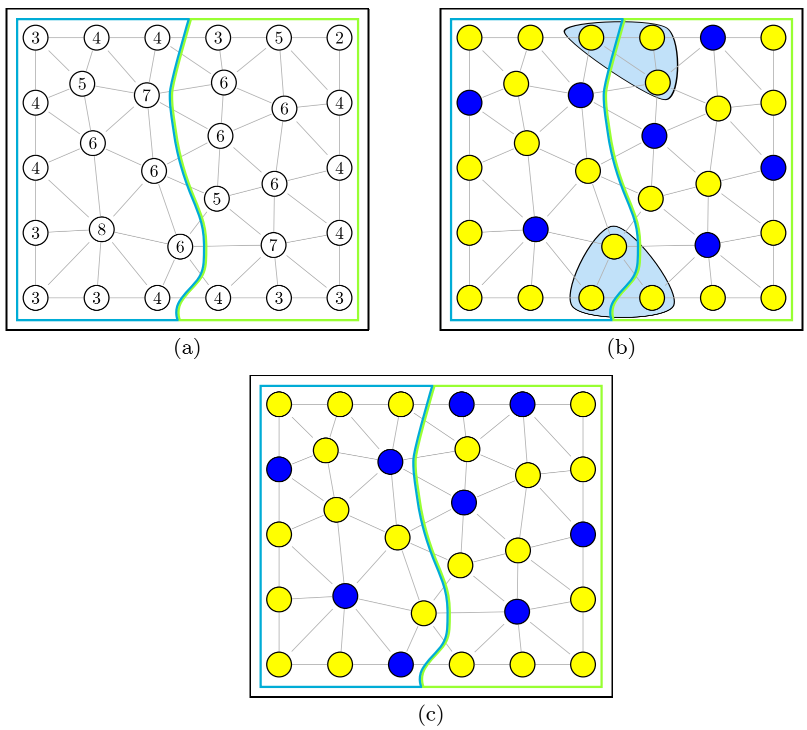


Figure 3.4: RS3 coarse-grid selection. Unassigned vertices are white, F-points are yellow, C-points are blue, and a processor boundary passes through the middle of the graph. Weight initialization is shown in (a). The coarse grid following the completion of RS on each processor is in (b), where violations of H1 are highlighted. Addition of new C-points yields the final coarse grid in (c).

Algorithm 3.2 Cleary-Luby-Jones-Plassmann (CLJP)

CLJP(S, $C = \emptyset$, $F = \emptyset$) {
for all $i \in V$ **do** /* ***initialization*** */
  $w_i \leftarrow |S_i^T| + \text{rand}[0, 1)$
end for
while $C \cup F \neq V$ **do** /* ***selection loop*** */
  $D \leftarrow$ Select-Independent-Set(S, w)
  $C \leftarrow C \cup D$
  for all $j \in D$ **do**
    Update-Weights(S, j, w)
    for all unassigned $k \in N_j$ **do**
      if $w_k < 1$ **then**
        $F \leftarrow F \cup \{k\}$
      end if
    end for
  end for
end while
return (C, F)
}

This set is guaranteed to be non-empty and independent, but not necessarily maximally independent.

Algorithm 3.3 CLJP Independent Set Selection

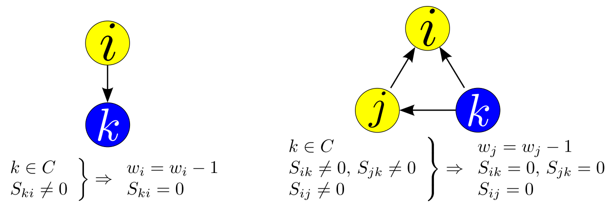
Select-Independent-Set(S, w) {
$D \leftarrow \emptyset$
for all $i \in V$ **do**
  if $w_i > w_j$ for all $j \in \mathcal{N}_i$ **then**
    $D \leftarrow D \cup \{i\}$
  end if
end for
return D
}

Weights are updated in CLJP using the following heuristics:

1. Values at C-points are not interpolated, so vertices that strongly influence a C-point are less valuable as potential C-points themselves. See Figure 3.5(a).

2. If i and j both strongly depend on $k \in C$ and j strongly influences i, then j is less valuable as a potential C-point since i can be interpolated from k and the influence of j on i can be interpolated through k. See Figure 3.5(b).

The implementation of these heuristics is outlined in Algorithm 3.4. In the algorithm, edges in the graph have a value of 1, the absence of an edge is represented by 0, and "removed" edges are represented by -1.



(a) (b)

Figure 3.5: Weight update heuristics used by CLJP.

Algorithm 3.4 CLJP Weight Update

```
UPDATE-WEIGHTS(S, k, w) {  /* k is new C-point */
for all j such that S_kj = 1 do  /* Heuristic 1 */
   w_j ← w_j − 1
   S_kj ← −1
end for
for all j such that S_jk ≠ 0 do  /* Heuristic 2 */
   S_jk ← −1
   for all i such that S_ij = 1 do
      if S_ik ≠ 0 then
         w_j ← w_j − 1
         S_ij ← −1
      end if
   end for
end for
}
```

An example of CLJP coarse-grid selection is shown in Figure 3.6.

3.4.3 Falgout

The principle drawback of parallel RS methods is the need to correct problems on processor boundaries, which often leads to poor behavior such as increased communication or unnecessary demands on memory. The quality of the coarse grid in the processor interior, however, is high. CLJP, on the other hand, selects coarse grids without needing to make corrections on processor boundaries, but produces poor results on structured meshes. The cause for this is the random augmentations used to initialize the vertex weights. This phenomenon is investigated in Chapter 4. Hybrid methods using a combination of RS and CLJP, therefore, are of natural interest and produce two strategies. One technique, called BC-RS [37], uses CLJP on processor boundaries followed by coarsening processor interiors using RS. The opposite approach – coarsening processor interiors followed by CLJP on processor boundaries – works more effectively. This method is called Falgout coarsening [37] and is outlined in Algorithm 3.5.

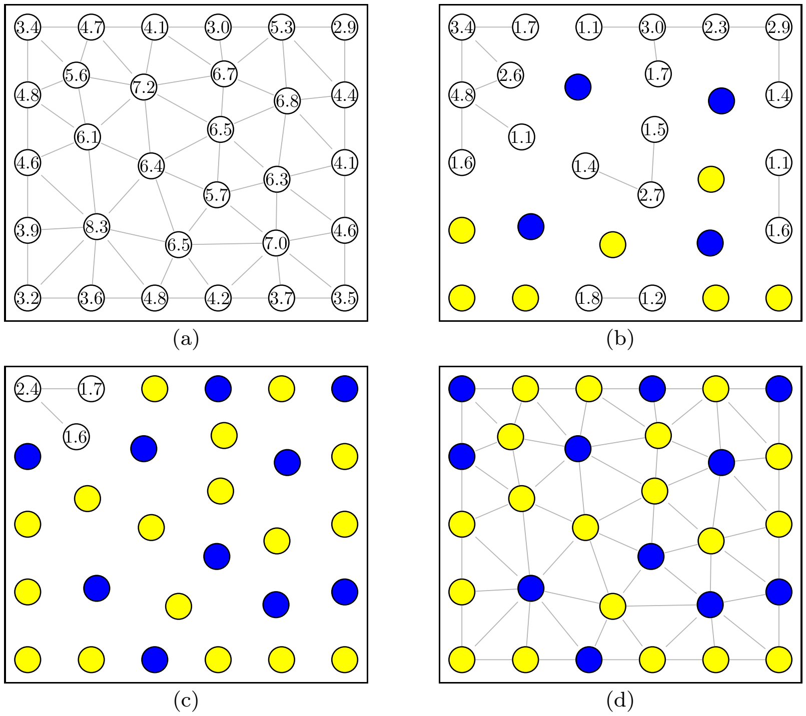


Figure 3.6: CLJP coarse-grid selection.

Algorithm 3.5 Falgout Coarse-Grid Selection

FALGOUT(S) {
  $(C, F) \leftarrow$ RUGE-STÜBEN(S)
  Mark processor boundary vertices as unassigned
  $(C, F) \leftarrow$ CLJP(S, C, F) /* *coarsen processor boundary with interior C/F assignments as input* */
  return (C, F)
}

The amount of memory needed to store the matrix operators from all levels is quantified as the *operator complexity.* The operator complexity is relative to the fine-level matrix and is defined as

$$C_{\text{op}} = \frac{\sum_{k=0}^{m} \text{nnz}_k}{\text{nnz}_0}, \tag{3.4}$$

where nnz_k is the number of nonzeros in A_k. Large operator complexities limit the number of degrees of freedom assigned to each processor, and operator complexity also influences the amount of work in a multigrid cycle since the cost of relaxation is proportional to the number of nonzeros.

Falgout provides a clear advantage and delivers for problems where CLJP fails to produce a

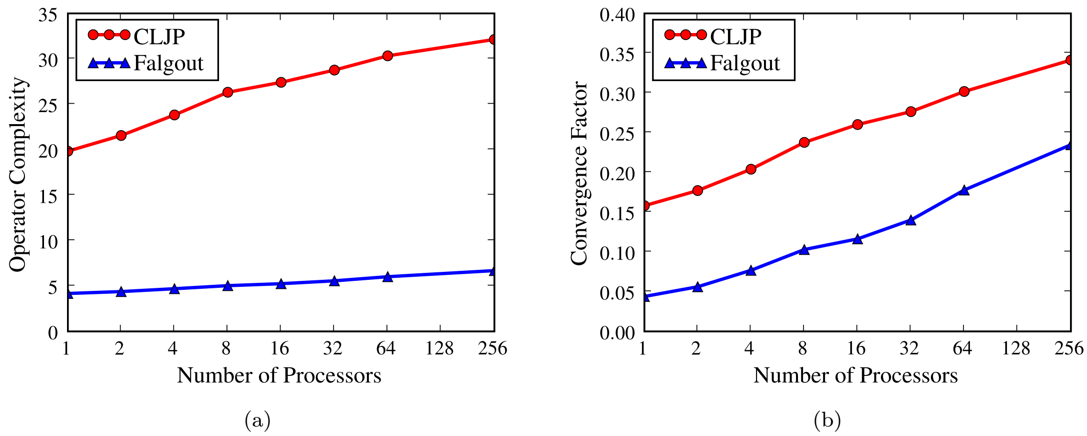


Figure 3.7: CLJP and Falgout coarsening on a 7-point Laplacian problem. Operator complexities are shown in (a), and convergence factors are shown in (b).

competitive AMG method and low operator complexities. Take the 3D Laplacian as an example:

$$\begin{aligned} -\Delta u &= 0 \quad \text{on } \Omega \qquad (\Omega = (0,1)^3), \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{3.5}$$

When (3.5) is discretized using finite differences to yield the common 7-point stencil, CLJP has particularly poor performance. Figure 3.7 plots the operator complexities and convergence factors for this problem when CLJP and Falgout are used as the coarsening algorithms. The domain in all tests is a $128 \times 128 \times 128$ grid, which gives approximately two million unknowns.

The operator complexity plot demonstrates the incredible difference in memory needs for the coarse-grid hierarchies produced by both methods. This impacts convergence factors and the time needed for relaxation.

For problems with unstructured grids, the differences between Falgout and CLJP become less pronounced. In some cases, CLJP yields a more effective AMG method [3].

3.5 H1′ Parallel Coarse-Grid Selection

For problems discretized on structured grids and 3D meshes, coarsening algorithms designed to satisfy H1 produce coarse-grid hierarchies with large operator complexities. Producing coarse-grid hierarchies with lower operator complexities motivated the development of additional algorithms using a modified coarsening heuristic called H1′:

H1′: For each F-point i, at least one $j \in S_i$ must be a C-point.

An equivalent statement is that each F-point must be strongly influenced by at least one C-point. In graph theory, a set satisfying this condition is a *converse dominating set* [31], and H1′ algorithms seek a minimal converse dominating set. The primary difference between H1 and H1′ is that H1′ does not require strongly connected F-points to share a strongly influencing C-point neighbor. This change allows for the selection of sparser coarse grids.

The algorithms presented below are designed to utilize the relaxed constraints of H1′ and are modified forms of CLJP and Falgout.

3.5.1 Parallel Modified Independent Set

Parallel Modified Independent Set (PMIS) [52] selects coarse grids using a modified form of CLJP. Like CLJP, PMIS utilizes Luby's maximal independent set algorithm to produce independent sets in each iteration, meaning that PMIS uses random weight augmentations when initializing vertex weights. Unlike CLJP, PMIS does not satisfy H1 and does not update vertex weights following the selection of C-points. A variant of PMIS, called PMIS Greedy [15], updates vertex weights following the selection of C-points in a fashion similar to that of RS. Algorithm 3.6 contains the implementation of PMIS, and an illustration of PMIS is shown in Figure 3.8.

Algorithm 3.6 Parallel Modified Independent Set (PMIS)

PMIS(S, $C = \emptyset$, $F = \emptyset$) {
for all $i \in V$ **do** /* ***initialization*** */
    $w_i \leftarrow |S_i^T| + \text{rand}[0, 1)$
end for
while $C \cup F \neq V$ **do** /* ***selection loop*** */
    $D \leftarrow$ Select-Independent-Set(S, w)
    $C \leftarrow C \cup D$
    for all $j \in D$ **do**
        for all $k \in S_j^T$ **do**
            $F \leftarrow F \cup \{k\}$
        end for
    end for
end while
return (C, F)
}

PMIS succeeds in selecting coarse-grid hierarchies with significantly lower operator complexities. The prolongation operator must be redefined for cases where strongly connected F-points do not share a strongly influencing C-point. The original fix for this situation is to treat the strongly influencing F-point neighbor as a member of the weakly connected neighbor set D_i^w instead of

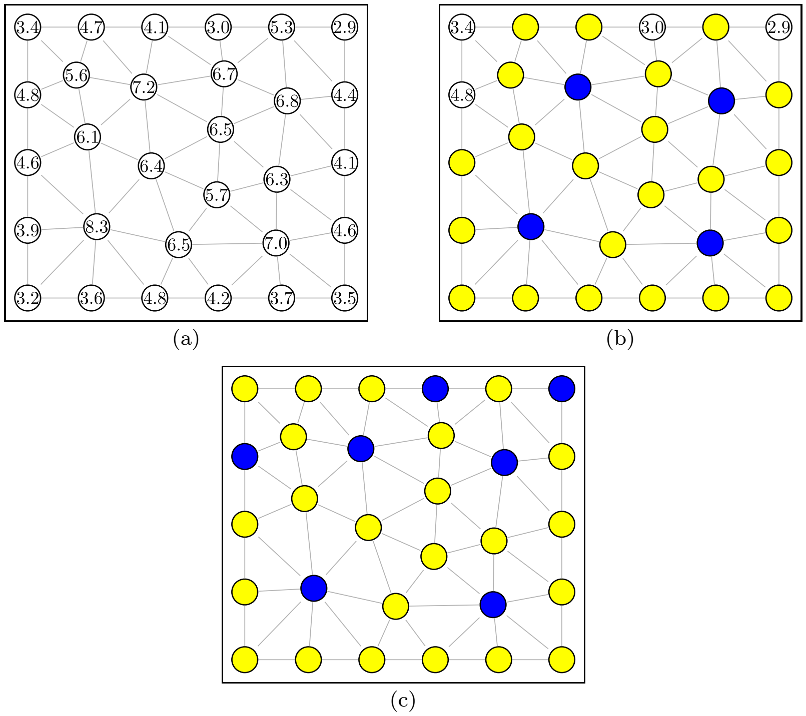


Figure 3.8: PMIS coarse-grid selection.

D_i^s [52]. Its influence is then treated in the same manner as other weakly connected neighbors in (2.16). This approach, however, is not accurate and often leads to poor convergence factors. Long-range interpolation offers more accuracy for H1′-selected coarse grids than standard interpolation and is currently an active area of research [15]. Additionally, the interpolation techniques in [10] are applicable to H1′-selected coarse grids.

3.5.2 Hybrid Modified Independent Set

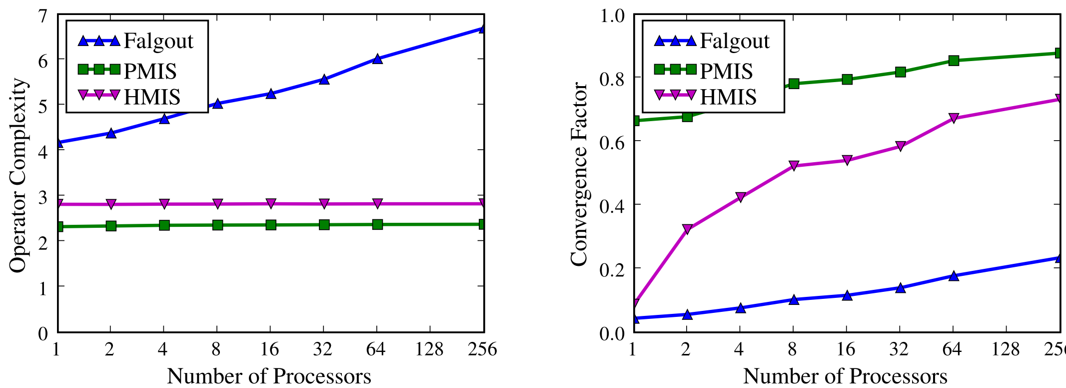
Similar to Falgout coarsening, Hybrid Modified Independent Set (HMIS) is a hybrid method. Rather than using CLJP on the processor boundaries, HMIS uses PMIS, while a first pass of RS is applied in the processor interiors. The first pass is sufficient since the second pass ensures strongly connected F-points share common strongly influencing C-points, which is not required for this method. HMIS is detailed in Algorithm 3.7.

3.5.3 Comparisons

To demonstrate the effect of PMIS and HMIS on operator complexity and convergence factors, consider the 3D 7-point Laplacian problem from Section 3.4.3. Figure 3.9 plots the experimental

Algorithm 3.7 Hybrid Modified Independent Set (HMIS)

```
HMIS(S) {
  (C, F) ← RUGE-STÜBEN(S) (first pass only)
  Mark processor boundary vertices as unassigned
  (C, F) ← PMIS(S, C, F)  /* coarsen processor boundary with interior C/F assignments as
      input */
  return (C, F)
}
```



(a) (b)

Figure 3.9: Falgout, PMIS, and HMIS coarsening on a 7-point Laplacian problem. Operator complexities are shown in (a), and convergence factors are shown in (b).

results for Falgout, PMIS, and HMIS. CLJP was not included in the plots in order to make operator complexity trends more visible.

PMIS and HMIS are designed to produce lower operator complexities, and the experimental results demonstrate smaller and more scalable operator complexities. Large convergence factors are typical for PMIS and HMIS when using the modified standard interpolation described in Section 3.5.1, demonstrating a need for improved interpolation operators for these methods. Despite poor convergence factors, the cost of HMIS is only around twice the cost of Falgout to gain a digit of accuracy in the residual, while the cost of PMIS is four times larger. This is due to cheaper relaxation sweeps in PMIS and HMIS compared to Falgout.

Chapter 4

Color-Based Parallel Coarse-Grid Selection

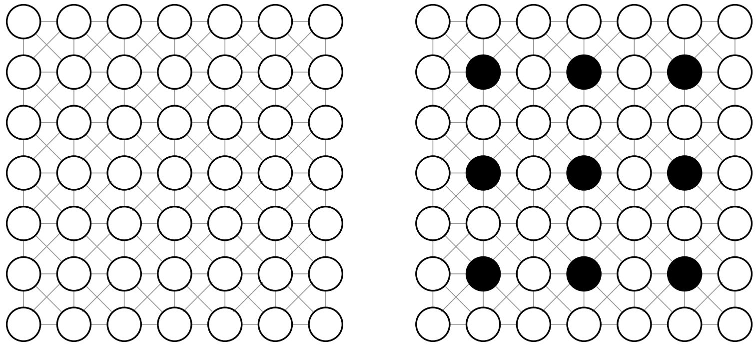
CLJP tends to select grid hierarchies with large operator complexities, particularly for structured two-dimensional or three-dimensional grids. Observations show that CLJP coarsening yields hierarchies with large complexities due to the nature of the independent sets used to select C-points. The independent sets in CLJP are constructed with the aid of random weight augmentations, which do not depend in any way on the mesh structure. As a result, the coarse grid typically does not retain the structured property of the fine grid. To address this phenomenon, the initialization of vertex weights is modified to encourage the selection of well-structured independent sets by using a graph coloring algorithm. Although the cost of coloring the graph prior to selecting the coarse grid is non-trivial, significantly fewer C-points are often selected, saving time.

This chapter introduces three algorithms designed to improve CLJP and its H1$'$ counterpart PMIS. Numerical experiments run using the modified CLJP algorithm, called "CLJP in color" (CLJP-c), show improvements in operator complexities for both structured 2D and structured 3D problems. In most cases, setup and solve times are markedly improved as well.

4.1 Analysis of CLJP

For certain types of problems, CLJP selects far more C-points than alternative algorithms, such as Falgout coarsening. Consider a small 9-point Laplacian problem with homogeneous Dirichlet boundary conditions, as in Figure 4.1(a). When the problem is coarsened using RS, the result is a structured coarse grid with nine vertices, as shown in Figure 4.1(b). When the 9-point Laplacian is coarsened with CLJP, the results are generally different since the random portion of the weight given to each vertex has significant control over the final coarse grid. As a result, CLJP selects a variety of grids.

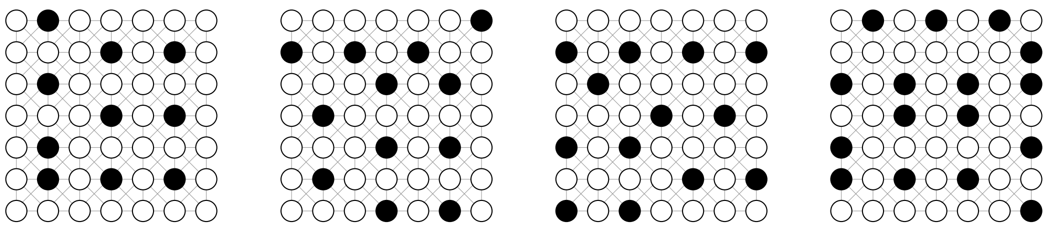
Since the randomly weighted elements do not take problem structure into account, structure is often lost on coarse levels. Figure 4.2 shows results from four selected runs by CLJP on the example



(a) 9-point Laplacian problem on a 7×7 grid.

(b) 9-point Laplacian problem coarsened using RS. C-points are represented as filled black circles.

Figure 4.1: A 9-point Laplacian problem and the results of coarsening with RS.



(a) 10 C-points. (b) 12 C-points. (c) 13 C-points. (d) 16 C-points.

Figure 4.2: Coarse grids selected by CLJP for a 9-point Laplacian problem on a 7×7 grid. C-points are represented with filled black circles.

problem. In each instance, more C-points are selected than by RS. This is not surprising since there is only one configuration satisfying heuristic H1 with nine C-points.

On structured meshes, CLJP generally selects coarse grids with significantly greater numbers of C-points than RS coarsening. Figure 4.3 plots observed frequencies of the number of C-points selected by CLJP for the 7×7 Laplacian across 250 million trials. CLJP selects coarse grids with nine C-points approximately two-percent of the time and selects between eleven and thirteen C-points sixty-percent of the time. In fact, coarse grids with nine C-points are the eighth most likely result.

These results are not limited to small problems. Figure 4.4 displays the same information as Figure 4.3 but for a larger problem: a 9-point Laplacian on a 512×512 grid. The mean number of C-points selected in 50,000 trials was 82,488. The minimum number selected was 82,210. For this problem, RS selects 65,536 C-points from a fine grid containing 262,144 vertices.

CLJP clearly selects too many C-points for certain types of problems. However, CLJP also exhibits several attractive qualities. Foremost, it is entirely parallel and the results do not depend on the processor topology. Also, the coarsening depends only on the weights assigned to vertices,

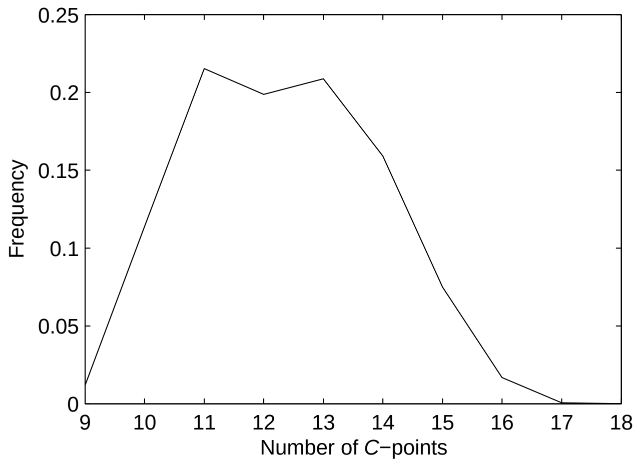


Figure 4.3: Experimental results from testing CLJP on the 9-point Laplacian in Figure 4.1(a). The plot shows the observed frequencies of the number of C-points selected by CLJP across 250 million trials.

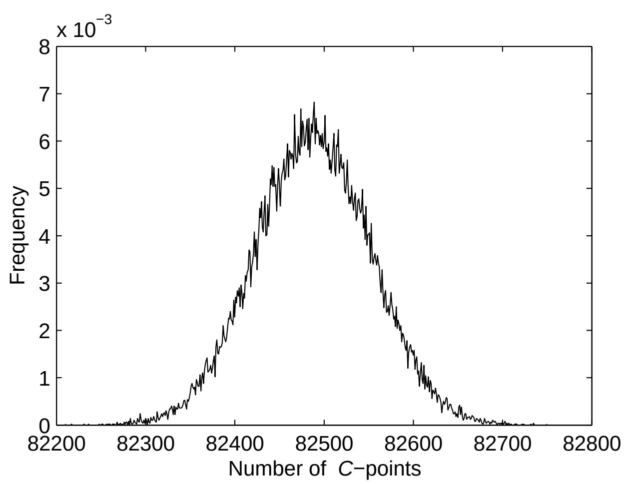


Figure 4.4: Experimental results from testing CLJP on the 9-point Laplacian on a 512×512 grid. The plot shows the frequencies of the number of C-points selected by CLJP in 50,000 trials.

meaning that CLJP selects the same coarse grid independent of the number of processors as long as the random weight augmentations are identical in both cases. This is beneficial for both AMG users and designers. Another positive quality in CLJP is that it coarsens to a single vertex without requiring data redistribution.

The ability to produce the same coarse grid regardless of processor topology and coarsening to a single vertex efficiently are qualities not shared by other RS-based coarse-grid selection algorithms. These attributes make it worthwhile to examine ways of improving the performance of CLJP.

4.2 Modifying CLJP

Methods to modify the CLJP algorithm in order to achieve better operator complexities for structured problems are discussed in this section.

4.2.1 Observations

Several observations follow from the experiments in Section 4.1 about the behavior of CLJP. First, Figure 4.2 demonstrates CLJP produces very different coarse grids when given different initial weights. Not all coarse grids perform equally well. However, CLJP does not consider structure when building a coarse grid. Further, the coarse grid selected affects how the solve phase performs and also affects the runtime of the setup phase.

CLJP is unlikely to select the same coarse grids as RS or Falgout coarsening, but it does have the ability to do so. The approach taken in this thesis is to develop a method that "encourages" CLJP to pick coarse grids similar to those selected by RS and Falgout coarsening. A benchmark goal for this method is to achieve performance comparable to, or better than, Falgout coarsening.

The results of CLJP coarsening depend significantly on the random weight augmentations of each vertex, so the initialization step is a good starting point for modification. The random weight augmentations exist to break ties between the weights of adjacent vertices. The negative effects of the augmentations on the coarse grid are unintended, so modifying the weights to preferentially select coarse grids with more structure is well-motivated.

4.2.2 Modifications

As discussed in the previous section, poorly structured independent sets in CLJP lead unstructured coarse grids. In this section, a modified CLJP algorithm is created by changing the weight initialization step (Line 2 in Algorithm 3.2) to include more information about the graph structure.

Graph coloring routines are used to implicitly extract information about the graph and create a more structured independent set. The graph coloring problem is as follows:

> Given a symmetric graph $G = (V, E)$, where V is the vertex set and E is the edge set, the graph coloring problem is to find a function σ which maps a color to each vertex such that $\sigma(v) \neq \sigma(w)$, for all $(v, w) \in E$.

The graph coloring operates on the *symmetrized strength matrix*. This is simply the strength matrix modified so that all directed edges are replaced with undirected edges in the graph. Another view is that vertex i has a different color from all $j \in \mathcal{N}_i$.

Finding the optimal graph coloring (i.e., the graph coloring with the minimum number of colors) is an NP-complete problem [32]. However, many heuristics to find near-optimal solutions have been created for this problem. For the coarse-grid selection problem, we are interested in creating a structured independent set, and many graph coloring heuristics satisfy this requirement.

Two types of coloring algorithms have been tested in this thesis: sequential greedy algorithms and Jones-Plassmann parallel algorithms, although any coloring heuristics are applicable for use with the algorithms introduced below. Sequential heuristics are able to compute graph colorings very close to optimal for a variety of graphs [54, 40]. Furthermore, given the appropriate vertex coloring order, an optimal coloring is obtained [6]. Greedy algorithms select a certain vertex coloring and typically deliver acceptable results. Two popular greedy heuristics are *saturation degree ordering (SDO)* introduced in [12] and a modified SDO called *incidence degree ordering (IDO)* [54]. In SDO, the next vertex selected to be colored is the vertex adjacent to the greatest number of colors in the graph. In IDO, the next vertex to be colored is adjacent to the greatest number of colored vertices. The method introduced below uses an IDO ordering for coarse-grid selection for some simulations, as in the case in the experiments at the end of this chapter. Lexicographic ordering (i.e., an ordering where the vertices are colored in the order of their enumeration), however, is often more effective since vertices in finite element meshes typically have a consistent, nearest-neighbor type of enumeration. The lexicographic approach is less robust and requires the vertices to be enumerated in a regular pattern, which is frequently the case.

The initial weights are modified to incorporate graph colors, which encourages CLJP to preferentially select certain colors for inclusion on the coarse grid. In Figure 4.5, a 9-point Laplacian on a 7×7 grid is coarsened using the modified CLJP algorithm, called "CLJP in color" (CLJP-c) [2]. Although the resulting coarse grid is the same set as the black vertices in the middle graph, this

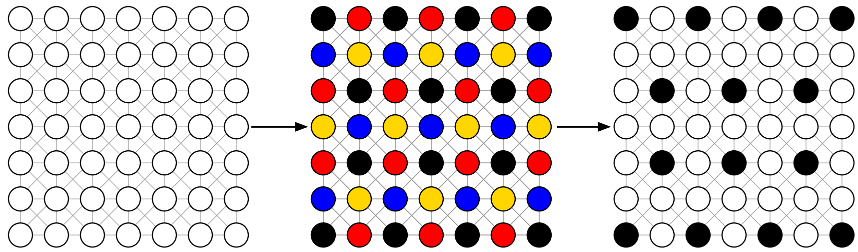


Figure 4.5: Outline of CLJP-c. In the first step, CLJP-c takes the graph of the strength matrix (left) and colors each vertex such that the color of vertex i is different than the color of all $j \in \mathcal{N}_i$ (middle). In the next step, a modified initialization phase is run such that each color carries some weight and there is a prioritization of colors. The result (right) shows the coarse grid selected by CLJP-c when the black vertices are given priority. For this simple problem, the C-point set is the set of black vertices.

situation is not typical. Each color in the graph is guaranteed to be an independent set, but a maximal independent set is necessary to satisfy the first condition of H1. Maximal independent sets are not, however, sufficient for satisfying the second condition of H1 for many graphs. In Figure 4.5 the black vertices are a maximal independent set.

In CLJP, the weight for vertex i is initially the sum of the number of strong influences of i and a random number in $(0, 1)$. The CLJP-c initialization includes information about the color of each vertex. The CLJP-c initialization is outlined in Algorithm 4.1.

Algorithm 4.1 Weight Initialization for CLJP-c

CLJP-C-WEIGHT-INITIALIZATION(S) {
1: $\sigma \leftarrow$ COLOR-GRAPH(S) /* *$\sigma(i)$ is color of vertex i* */
2: $c_\ell \leftarrow$ set of colors in $G(A)$ /* *"colors" are sequentially numbered integers beginning with 1* */
3: **for all** colors $c \in c_\ell$ **do**
4: $\quad colorWeight(c) \leftarrow (c-1)/|c_\ell|$
5: **end for**
6: **for all** $i \in V$ **do**
7: $\quad w_i \leftarrow |S_i^T| + colorWeight(\sigma(i))$
8: **end for**
}

New to the CLJP-c initialization is the *color weight*. The color weight is a unique weight given to each color and is included in vertex weights during initialization. The purpose is to establish a hierarchy among colors. For example, the middle graph of Figure 4.5 has four colors, so the set of colors $c_\ell = \{1, 2, 3, 4\}$. The color weights are 0.0, 0.25, 0.5, and 0.75. In Figure 4.5, the black vertices are color 4 and therefore have the "largest" weight among the colors. Line 4 of Algorithm 4.1 leads to augmentations of 0.75 to the weights of black vertices, which is a greater augmentation than any of their neighbors. Thus interior black vertices become C-points in the first iteration of the loop, and the remaining black vertices are selected in the second iteration of the loop.

4.2.3 Parallel CLJP-c

The choice of graph coloring algorithm impacts the overall design of parallel CLJP-c. The research here explores three techniques for the implementation of CLJP-c. These methods use different techniques to ensure no ties in the vertex weight exist across processor boundaries.

Ideally, a well-structured graph coloring is produced by a parallel graph coloring algorithm. The parallel graph coloring algorithm tested with CLJP-c [39] does not produce such results. This parallel coloring algorithm works by first coloring processor boundaries in parallel and then by coloring processor interiors using a sequential coloring algorithm with the processor boundary coloring as input. A consistent graph coloring (i.e., one that satisfies the graph coloring heuristic for all vertices, regardless or processor assignments) is produced, but leads to "tears" in the graph coloring. The tears are a result of approaching color fronts on processor interiors. When the fronts meet, segments exist where the coloring does not match and an out-of-place color is assigned. The strips of miscoloring have a small impact on the number of colors in the graph, but have a significant impact on the structure of the coloring, which negatively affects the improvements provided by CLJP-c.

The next two approaches utilize a sequential graph coloring algorithm that produces a structured coloring in processor interiors. A side-effect of using sequential coloring is inconsistent coloring along processor boundaries. To handle inconsistencies, one approach utilizes the CLJP method of using random augmentations to vertex weights. In the second approach, a parallel graph coloring is employed along processor boundaries, in addition to the sequential coloring, to break any ties between processor boundary vertices. In both approaches, vertices are guaranteed to have unique weights within their neighborhoods.

4.2.4 Implementation Specifics

The algorithms used for the experiments of Section 4.5 are developed within the *hypre* framework [28] from the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory.

The graph coloring algorithm used is a greedy algorithm with IDO as the ordering. The IDO algorithm is straightforward to implement, but does have shortcomings. In particular, the greedy algorithm blocks CLJP-c from producing the same coarse grid invariant of processor topology.

4.3 PMIS-c1 and PMIS-c2

The former sections examine a color-based modification to CLJP to improve performance. Recall from Figure 3.1 that CLJP is the source of most parallel independent set-based coarsening algorithms today. CLJP and PMIS are very similar in design, so the techniques in CLJP-c apply equally well to PMIS. As a result, two H1′ color-based coarsening algorithms are introduced: PMIS-c1 and PMIS-c2 [3].

PMIS-c1 is a direct application of CLJP-c for H1′. That is, the reduced graph of strong connections is colored such that no two adjacent vertices are assigned the same color. A color priority is formed and weights in the initialization phase of PMIS are modified so that color priority is enforced. Consequently, coarse grids emerge that are similar to those produced by HMIS. The algorithm is generated by replacing Line 1 and Line 2 of Algorithm 3.6 with Algorithm 4.1.

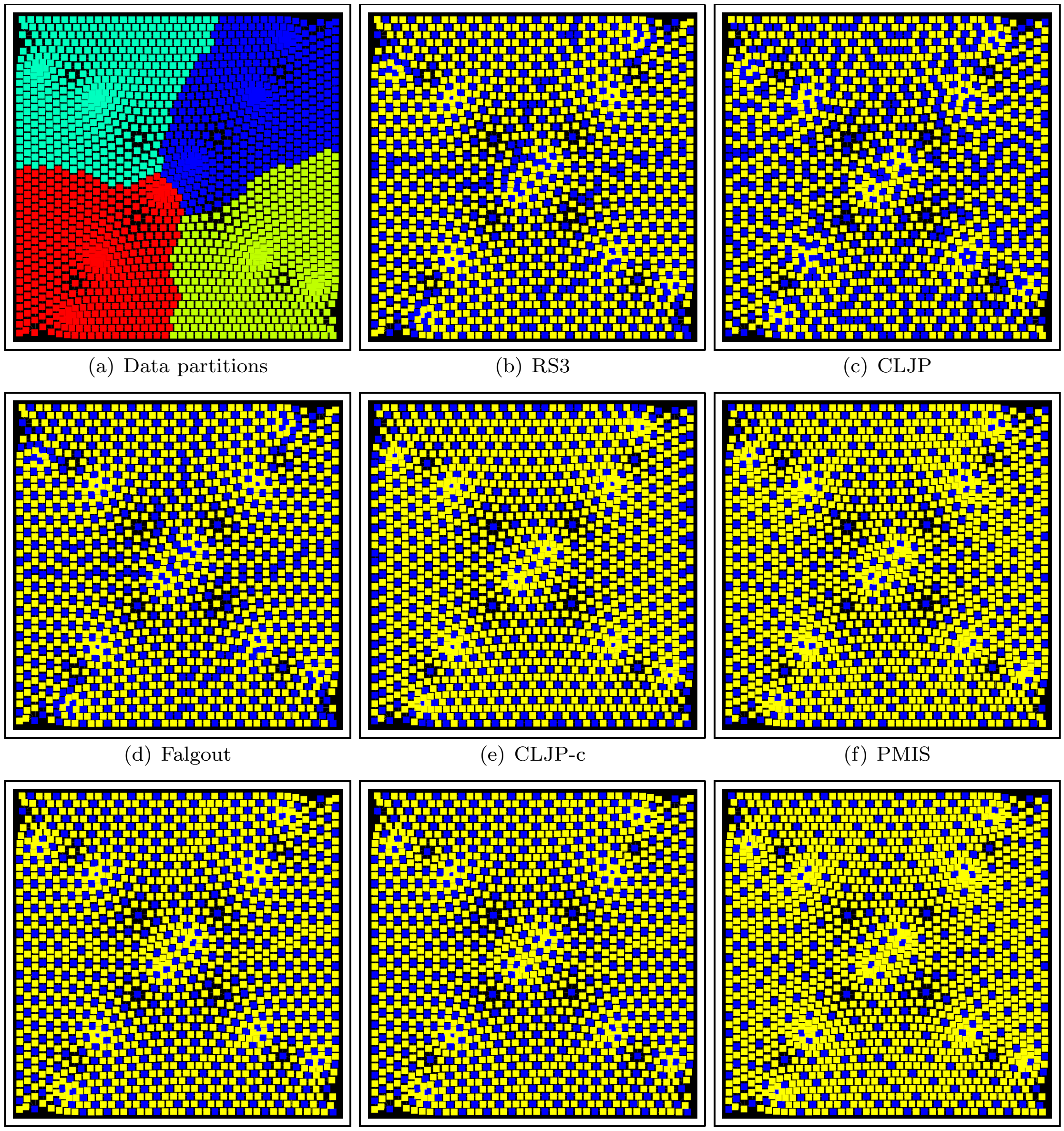
PMIS-c2, on the other hand, is developed from the observation that PMIS is able to (and does, given appropriate weights) select a coarse grid such that C-points are three hops from other C-points in the graph, when taken to the limit. To produce a coarse grid with a distance-three sparsity pattern using the graph coloring framework, a distance-two coloring algorithm is used. Now each vertex is assigned a color that is unique from the colors of all vertices within two hops in the graph of the symmetrized strength matrix. After substituting the graph coloring algorithm, PMIS-c2 proceeds similar to PMIS-c1. The algorithm is identical to PMIS-c1, with the exception that Line 1 of Algorithm 4.1 should read as follows.

1: $\sigma \leftarrow$ Color-Graph-Distance-Two(S) /* σ_i *is color of vertex* i */

PMIS-c2 selects very aggressive coarse grids and requires accurate long-range interpolation in order to yield AMG methods with acceptable convergence factors. The value of PMIS-c2 improves as long-range interpolation methods continue improving.

4.4 Method Fingerprints

The purpose of this section is to build intuition regarding the coarse grids selected by the algorithms introduced in Sections 4.2 and 4.3. Consider a small square domain partitioned into four domains by ParMETIS [41]. The problem is discretized by finite elements with piecewise linear basis functions. Figure 4.6 shows the partitions and C/F splittings algorithms in this chapter and in Chapter 3 compute.



(a) Data partitions (b) RS3 (c) CLJP

(d) Falgout (e) CLJP-c (f) PMIS

(g) HMIS (h) PMIS-c1 (i) PMIS-c2

Figure 4.6: Coarse grids selected by various parallel coarsening algorithms for a discretized Laplacian problem on an unstructured mesh. Figure (a) shows the four processor domains. Yellow squares are F-points, and blue squares are C-points.

4.5 Experiments

The performance of parallel coarse-grid selection algorithms is studied computationally in this section.

4.5.1 Methods and Measures

Each problem is tested using seven coarsening algorithms: Falgout, CLJP, CLJP-c, PMIS, HMIS, PMIS-c1 and PMIS-c2. Runs are made with the number of processors ranging from one to 512 in powers of two. Strength threshold θ (see Equation 2.10) is 0.25 for all coarsening algorithms, and while changing θ may affect the complexities, the overall trends are expected to remain unchanged. Thunder, a large parallel machine at Lawrence Livermore National Laboratory, is used for all experiments. Thunder has 1002 quad-processor Itanium2 computing nodes, each with 8GB RAM.

Problem Generation and Partitioning

Problems on regular grids are generated using routines in *hypre* [29]. Load balancing and problem partitioning is not problematic since they are readily partitioned into portions of equal size.

The generation of problems on unstructured grids is done using the aFEM package [42], which is a scalable, unstructured finite element problem generator. aFEM uses ParMETIS to partition the problem domain prior to discretization. To test the behavior of AMG as problem size is scaled, the amount of work given to each processor should be comparable. Equal sized partitions, however, are not guaranteed for unstructured problems, so the amount of work assigned to each processor is monitored in each test. Where the partitioning significantly departs from what is desired, plots containing information on the partitioning of the problem are provided. For example, Figure 4.16 depicts the evolution of problem size through the experiment. First, the solid black line shows the average number of unknowns per processor and is ideally constant. Surrounding the line are two shaded fields representing the number of unknowns per processor. The dark gray field shows the range in the number of unknowns per processor for all processors, whereas the light gray field shows the range in the middle 90% of the distribution. Finally, the dashed line is drawn horizontally from the average vertices per processor on the single processor trial.

Grid and Operator Complexity

In AMG, complexities are used to measure the size of the coarse-grid hierarchy. Grid complexity is the number of unknowns (or vertices, in terms of the graph) on all levels relative to the fine level:

$$C_{\text{grid}} = \frac{\sum_{\ell=0}^{m} n_\ell}{n_0}, \tag{4.1}$$

where m is the number of levels in the grid hierarchy and n_ℓ is the number of unknowns in the matrix on level ℓ.

Recall from Section 3.4.3, operator complexity (3.4) is the number of nonzeros in the matrices on all levels relative to the nonzeros in the fine-level matrix:

$$C_{\text{op}} = \frac{\sum_{\ell=0}^{m} \text{nnz}_\ell}{\text{nnz}_0}, \tag{4.2}$$

where nnz_ℓ is the number of nonzeros in the matrix on level ℓ. The operator complexity is a measure of the amount of memory needed, relative to the fine level, to store all of the matrices. It is also a lower bound on the computation needed since the cost of relaxation depends on the number of nonzeros in the matrices.

Convergence Factors

Convergence factor results provide information about the overall quality of the solve phase. For results in this section, convergence factors are computed by averaging convergence factors from all iterations until the norm of the relative residual is smaller than 10^{-8}. If a relative residual norm of 10^{-8} is not attained within 100 iterations, convergence factors from the first 100 iterations are averaged.

Work per Digit-of-Accuracy

Neither convergence factor nor operator complexity results alone measure the amount of work required by a solve phase. By combining the convergence factor and cycle complexity, a measure of the amount of work needed per digit-of-accuracy is realized. Work per digit-of-accuracy is defined as

$$W_{\text{digit}} = -\frac{C_{\text{cycle}}}{\log \rho}, \tag{4.3}$$

where ρ is the convergence factor and cycle complexity is a measure of work in each multigrid cycle. The cycle complexity is related to the operator complexity and is defined as

$$C_{\text{cycle}} = \frac{\sum_{\ell=0}^{m} \text{nnz}_\ell \cdot \nu_\ell \cdot \gamma^\ell}{\text{nnz}_0}, \tag{4.4}$$

where nnz_ℓ is the number of nonzeros in the matrix on level ℓ, ν is the sum of the number of presmoothing and postsmoothing steps on level ℓ, and γ is the cycle index. Recall the cycle index is

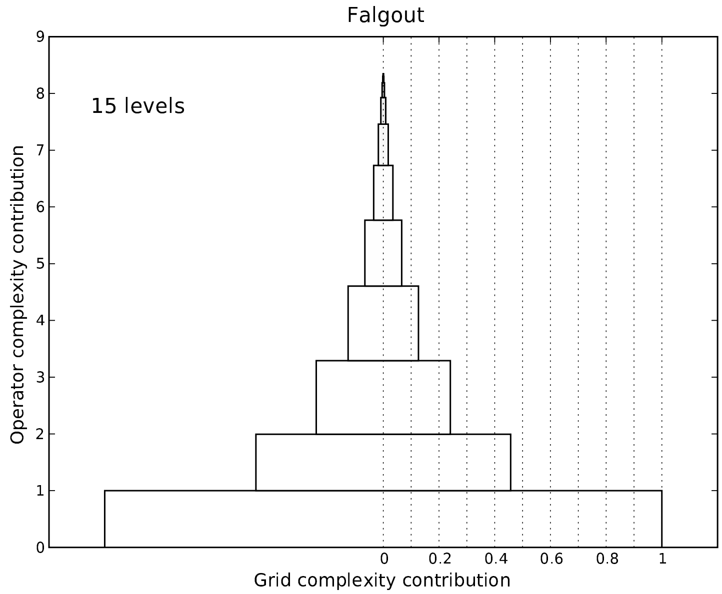


Figure 4.7: An example tower plot. The tower plot represents information about the coarse-grid hierarchy. Each level in the tower represents a level of the grid hierarchy, with the bottom level being the finest level. Four pieces of information are represented in each level of the tower: operator complexity, grid complexity, operator density, and the number of levels in the grid hierarchy.

used in the definition of the multigrid cycle (see Section 2.3.2). All experiments herein use a V(1,1) cycle, which implies the cycle complexity is approximately double the operator complexity.

Tower Plots

To visualize and examine the properties of the grid hierarchy in more detail, a new visualization tool called the tower plot is introduced. Tower plots visualize the entire coarse-grid hierarchy, level by level, as illustrated in Figure 4.7. Each tower plot contains four pieces of information. First, the height of a rectangle is that level's contribution to operator complexity. For example, the height of level ℓ is $\mathrm{nnz}_\ell/\mathrm{nnz}_0$. The total height of the tower is the total operator complexity of the hierarchy. Second, the width of each level corresponds to that level's contribution to grid complexity (i.e., the number of degrees of freedom on that level relative to the fine level). The grid complexity for a level is read by determining the location of the right edge of the corresponding block. For example, the third level in the grid level hierarchy in Figure 4.7 contains approximately 24% of the number of degrees of freedom of the fine level. Third, the darkness of the corresponding rectangle's fill color represents the sparsity of the matrix on level ℓ. In most cases, rectangle color remains white until the coarsest levels. Finally, the coarsening algorithm and the number of levels in the grid hierarchy is listed.

4.5.2 Fixed Problem Sizes

The experiments are divided into two broad types of tests. The total size of the problem is fixed in the first set of tests, regardless of the number of processors used. The purpose is to demonstrate the behavior of the coarsening algorithms on processor boundaries. By keeping the size of the problem fixed, the "surface area" of each processor domain increases as the number of processors is increased. This is not a natural test for performance scalability; in a real-world simulation, the experimenter normally assigns each processor an optimal amount of work.

3D 7-point Laplacian

The first problem is a 3D Laplacian:

$$\begin{aligned} -\Delta u &= 0 \quad \text{on } \Omega \qquad (\Omega = (0,1)^3), \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{4.5}$$

The problem is discretized using finite differences to yield the common 7-point stencil. The domain in all tests is a $128 \times 128 \times 128$ grid, resulting in approximately two million unknowns.

Some coarsening algorithms are more sensitive to processor boundaries than others, so degradation in performance as the number of processors increases is expected. The algorithms most sensitive to processor boundaries are the hybrids (Falgout, HMIS) and the graph coloring-based algorithms (CLJP-c, PMIS-c1, PMIS-c2).

Figure 4.8 plots setup times, convergence factors, operator complexities, and work per digit-of-accuracy for each of the trials in the experiment. The large operator complexities of CLJP makes it difficult to see the behavior of the others in detail, so a second plot of operator complexities is displayed in Figure 4.9. The overall trend is a decrease in setup time of each coarsening algorithm as the number of processors is increased. CLJP experiences the greatest performance gains as the number of processors grows, but requires large amounts of work to build the coarse-level hierarchy. The amount of work saved by splitting work across processors is much larger than the cost in communication. On the other hand, several coarsening algorithms initially experience an increase in setup time due to the minimal amount of work. Time spent communicating for the two processor test is not offset by savings in computing time, so total time increases. In the limit, however, all of

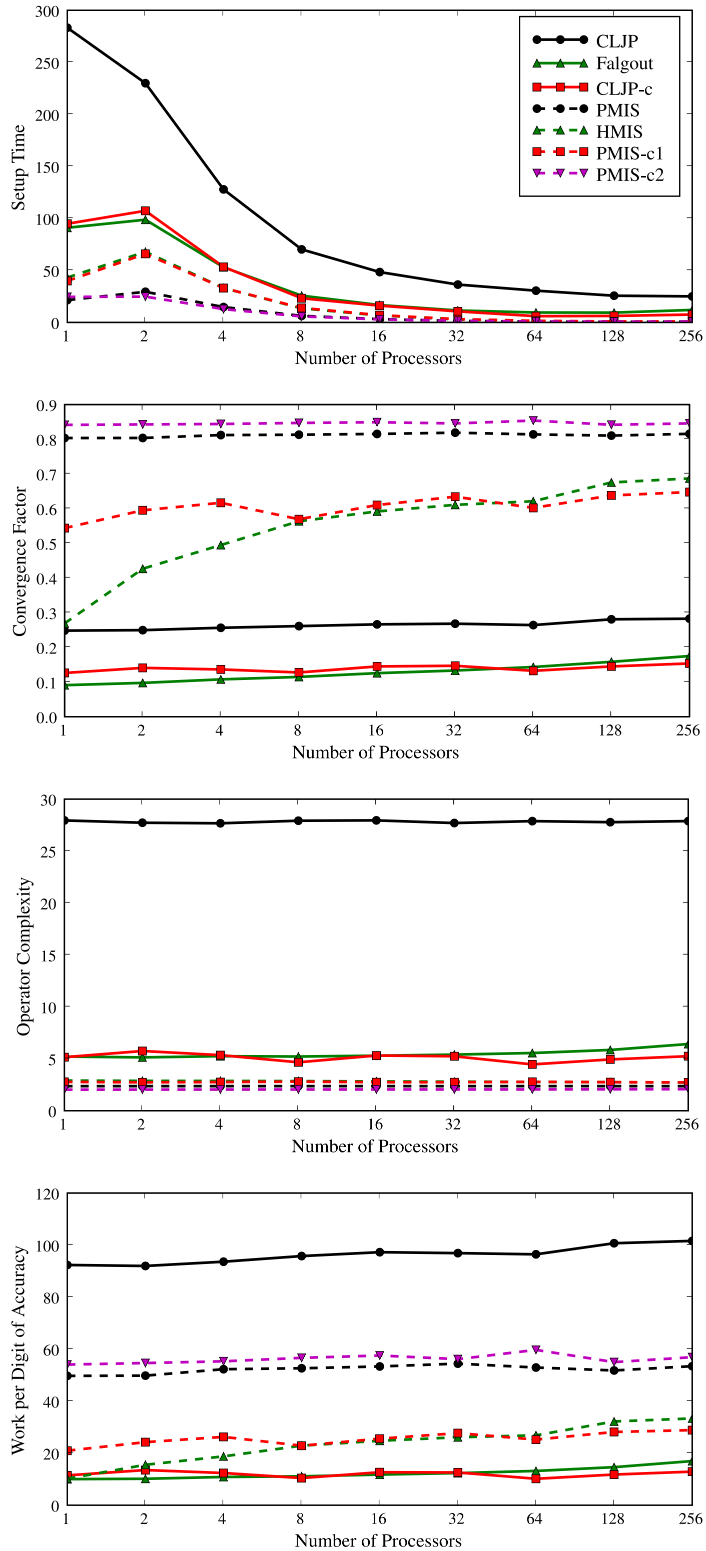


Figure 4.8: Results for the fixed problem size 3D 7-point Laplacian problem. The total degrees of freedom in the problem is fixed while the number of processors increases. The legend from the first plot applies to all four plots.

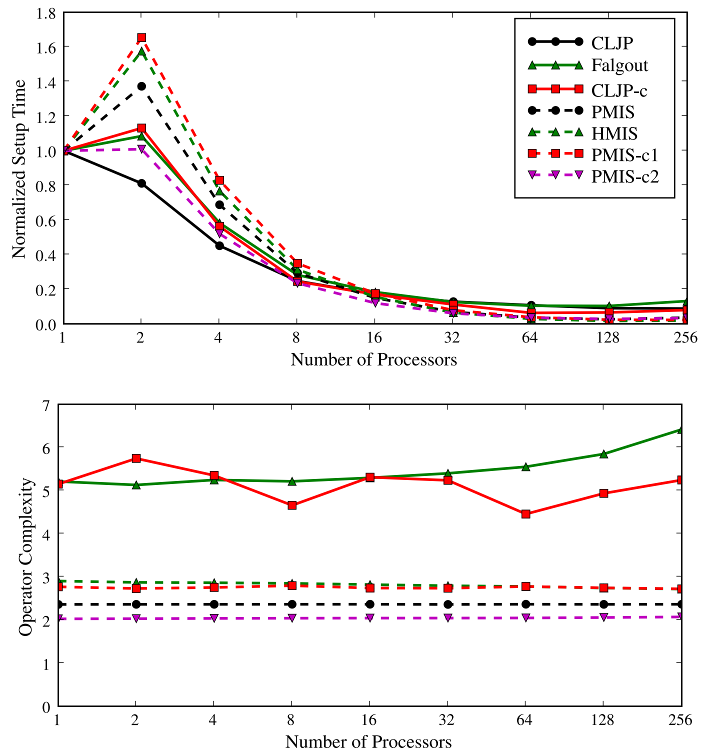


Figure 4.9: Normalized setup times and a closer view of operator complexities for the fixed problem size 3D 7-point Laplacian problem.

the algorithms experience decreases in setup time.

There is a practical limit to gains made through parallelism for coarse-grid selection algorithms due to communication across processor boundaries. Figure 4.9 shows *normalized setup time*, which is the setup time in a trial divided by the setup time in the single processor trial. At the right side of the plot, some lines are increasing, meaning the savings in computation are no longer larger than the extra cost in communication. Additionally, the setup phase requires the least amount of time on 128 processors, where the times are between 2% and 10% of the times on a single processor.

The preferred outcome for the convergence factors is invariance with the number of processors since parallelism does not improve the rate of convergence, but rather targets the computational cost in each iteration. In most cases, convergence factors are constant across all trials; the largest exception is HMIS.

The increase in convergence factors for HMIS is due to large differences in the "quality" of the coarse grid in the interior and the coarse grid on the processor boundary for HMIS. The interior part of each processor's portion of the mesh in HMIS is coarsened similarly as in RS, which performs at least as well as Falgout in terms of convergence factor. However, the processor boundary coarse grid is selected using PMIS. The plot shows the PMIS convergence factors are diminished. As the number of processors is increased, the HMIS coarse grids have more vertices coarsened by PMIS, so performance degrades.

Similar to convergence factor, it is desirable for increased parallelism to have little impact on the operator complexity as the problem is divided among multiple processors. Both CLJP and PMIS are largely immune to the number of processors since they have the ability to produce the same coarse-grid hierarchy independent of the number or processors on which the algorithm is run [18, 52]. The operator complexity analyzed in Figure 4.8 shows each algorithm produces coarse-grid hierarchies of similar operator complexities on one processor versus hundreds of processors. The largest increase occurs with Falgout, which grows from approximately 5 to 6.5. In some cases, the operator complexity decreases by a small amount as the number of processors increased. Finally, as expected from previous observations [37, 52, 2], CLJP produces grid hierarchies with operator complexities too large to be a viable method. CLJP produces unusually large operator complexities for problems on structured meshes, but this phenomenon is not present on unstructured meshes.

Recall work per digit-of-accuracy is a quantity that depends on both the cycle complexity and on the convergence factor. As shown in Figure 4.8, Falgout and CLJP-c are the most cost effective methods for the fixed problem size 7-point Laplacian. Falgout and CLJP-c exhibit the lowest convergence factors and also maintain moderate operator complexities compared to the lowest operator complexities observed. Despite a reasonable convergence factor, AMG with CLJP is more expensive than all other methods due to extremely large operator complexities.

PMIS, HMIS, PMIS-c1, and PMIS-c2 produce much lower operator complexities. The tower plots for each coarsening algorithm run on 256 processors are shown in Figure 4.10. On 256 processors, CLJP selects coarse grids that produce matrices with a larger number of nonzeros on levels 2–10 than on level 1. Level 10 has more nonzero entries than level 1, despite having less than 5% the number of unknowns. The tower plots illustrate the similarity in complexities of the grid hierarchies selected by HMIS and PMIS-c1 and reveal that these coarsening algorithms appear nearly identical in terms of operator and grid complexity. This is further emphasized by the plots in Figure 4.8, which show HMIS and PMIS-c1 are producing AMG solve phases with similar performance.

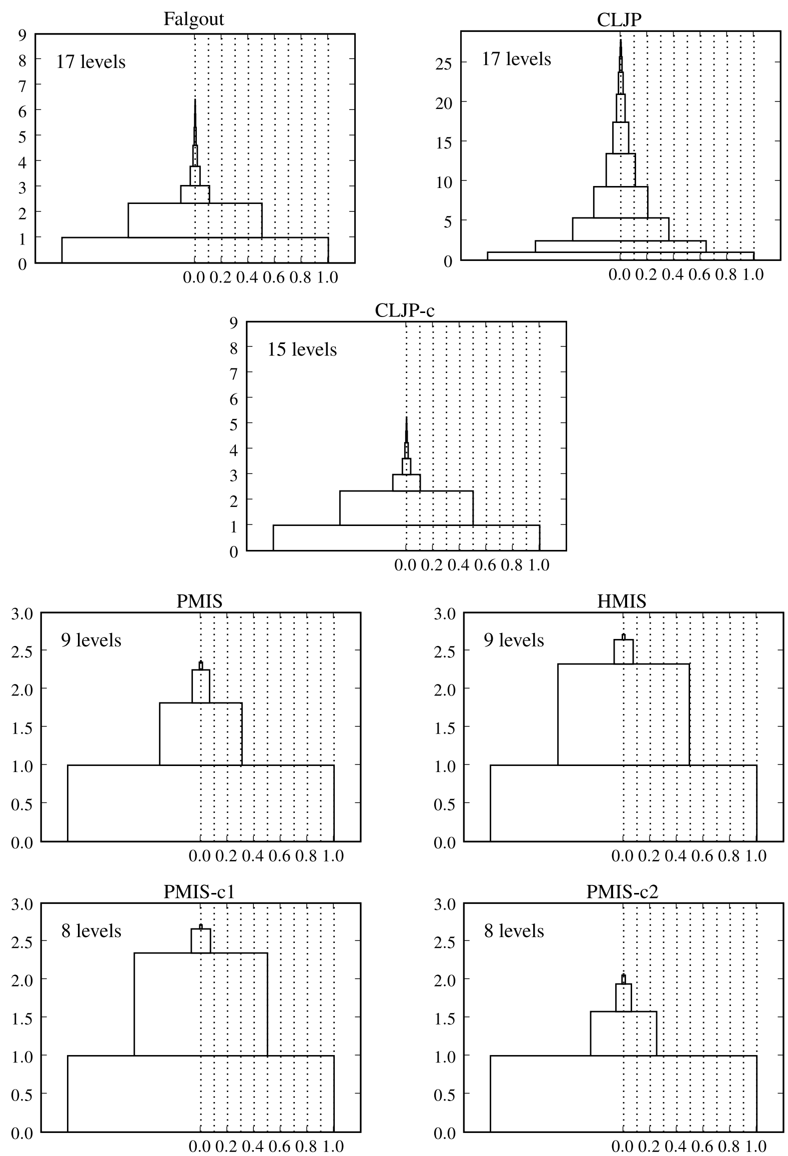


Figure 4.10: Tower plots for the fixed problem size 3D 7-point Laplacian problem. The towers shown are for the 256 processor trials. Notice the scale is not the same in each plot.

3D Unstructured Laplacian

The 3D unstructured Laplacian is also tested using a fixed problem size by varying the number of processors used to solve the problem. The Laplacian on the unit cube (4.5) is used, but is now discretized using finite elements on an unstructured mesh. The simulation is scaled up to 512 processors and has approximately 940,000 degrees of freedom. Figure 4.11 plots the setup time, convergence factor, operator complexity, and work per digit-of-accuracy results.

The setup times are relatively low compared to the structured case. The order of the algorithms by setup time is similar to the structured test, yet some differences are notable (e.g., CLJP is less expensive than both Falgout and CLJP-c in the problem). As before, the setup time reaches a minimum as parallelism is increased before beginning to increase after 128 processors. Figure 4.12 shows that the algorithms reach 10% of their single processor cost when run on 128 processors, in the worst case. Finally, the order of the algorithms by normalized setup time in Figure 4.12 is much different from the order in the structured case presented in Figure 4.9.

The convergence factors and operator complexities exhibit little variance as the problem is partitioned into more subdomains. Initially, there is growth in operator complexity when moving from one processor to two. Subsequently, operator complexities remain nearly constant. Operator complexities are much lower for CLJP in this experiment compared to the structured case, as depicted in Figure 4.13.

Much less work is needed per digit-of-accuracy in the unstructured test. In most cases, the amount of work per digit-of-accuracy is growing slightly, but the growth is relatively small given the number of processors used in the largest test.

The two fixed size tests are designed to explore the parallel behavior of the setup phase while using a variety of coarse-grid selection algorithms. The performance of AMG is largely insensitive to the number of processors used for these problems. Moreover, the operator complexities in AMG show little change regardless of the number of processors, even for the structured problem, which is highly impacted by coarse grids that do not maintain the structure of the fine grid. If a sufficiently large number of processors is used, operator complexities are expected to degrade, but 512 processors is already a departure from practical conditions for a problem of this size.

4.5.3 Scaled Problem Sizes

The fixed size tests from the previous section are designed to illustrate how the coarse-grid selection algorithms work as parallelism is increased. In practice, however, it is expected that as few processors

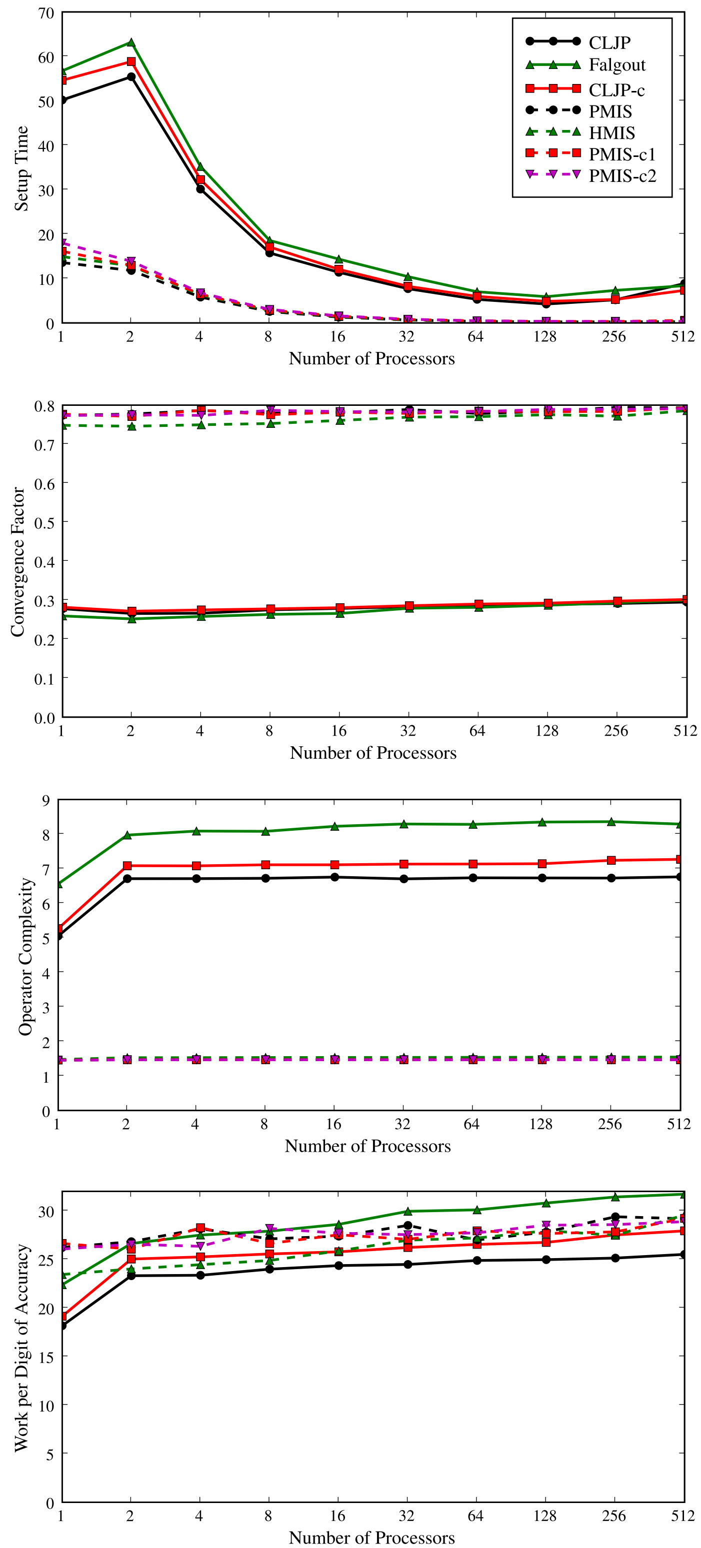


Figure 4.11: Results for the fixed problem size 3D unstructured Laplacian problem discretized on the unit cube. The total degrees of freedom in the problem is fixed while the number of processors increases. The legend from the first plot applies to all four plots.

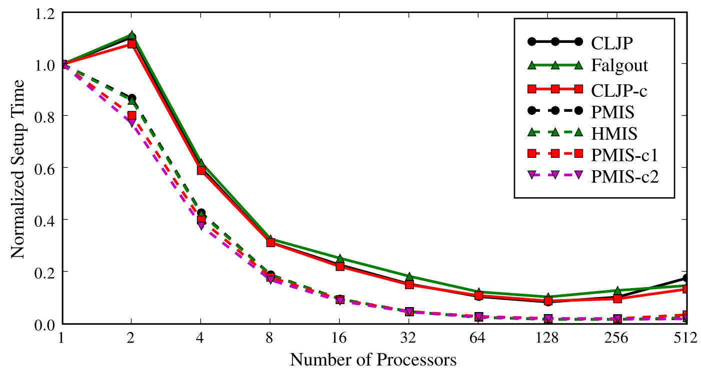


Figure 4.12: Normalized setup times for the fixed problem size 3D unstructured Laplacian problem discretized on the unit cube.

as necessary are used to solve a given problem.

The remainder of the experiments in this thesis scale the size of the problem to match the number of processors used. That is, the number of unknowns per processor is kept close to the number of unknowns on a single processor. This allows the setup phase algorithms to be observed under more natural conditions.

3D 7-point Laplacian

The structured problem (4.5) is now re-addressed, except the problem size is scaled as the number of processors increases. On one processor, the problem is discretized on a $50 \times 50 \times 50$ grid, for a total of 125,000 unknowns. On 256 processors, the problem is on a $400 \times 400 \times 200$ grid, which results in 32 million unknowns. Such small problem sizes are necessary for the algorithms producing high operator complexities to have sufficient memory. The results for normalized setup time, convergence factor, operator complexity, and work per digit-of-accuracy are given in Figure 4.14. The plots reveal very different results than the plots of Section 4.5.2.

The figure illustrates that some algorithms are not performing near optimal in terms of setup time. In particular, CLJP, CLJP-c, and Falgout are each exhibiting large growths in their setup times. CLJP setup time is growing more slowly than Falgout and CLJP-c, but the growth is still significant. In terms of actual time (see Section C.3), CLJP is more expensive than CLJP-c or Falgout on 256 processors, but assuming the trend continues, CLJP requires less time than Falgout and CLJP-c for the problem run on 1024 processors. Interestingly, the operator complexities of the grid hierarchies generated by CLJP are extremely large, creating large numbers of edges in the coarse level graphs, which requires large amounts of time for CLJP to update vertex weights.

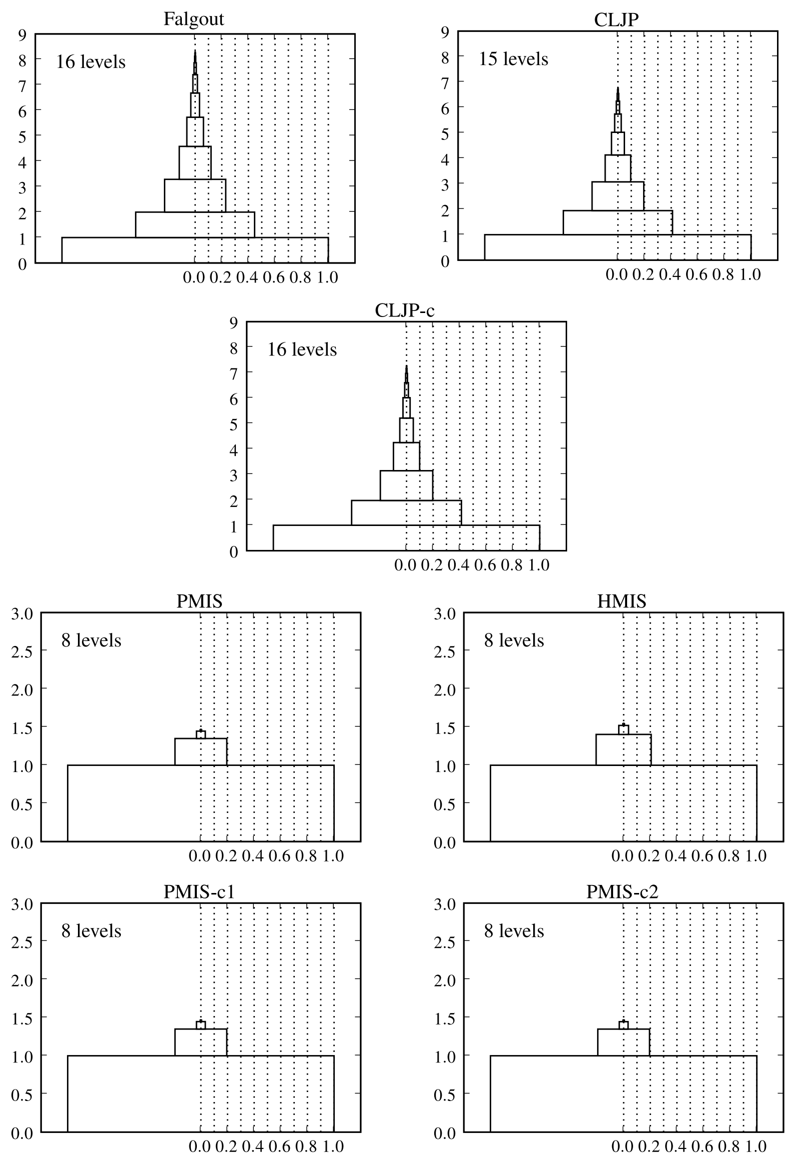


Figure 4.13: Tower plots for the fixed problem size 3D unstructured Laplacian problem on the unit cube. The towers shown are for the 512 processor trials.

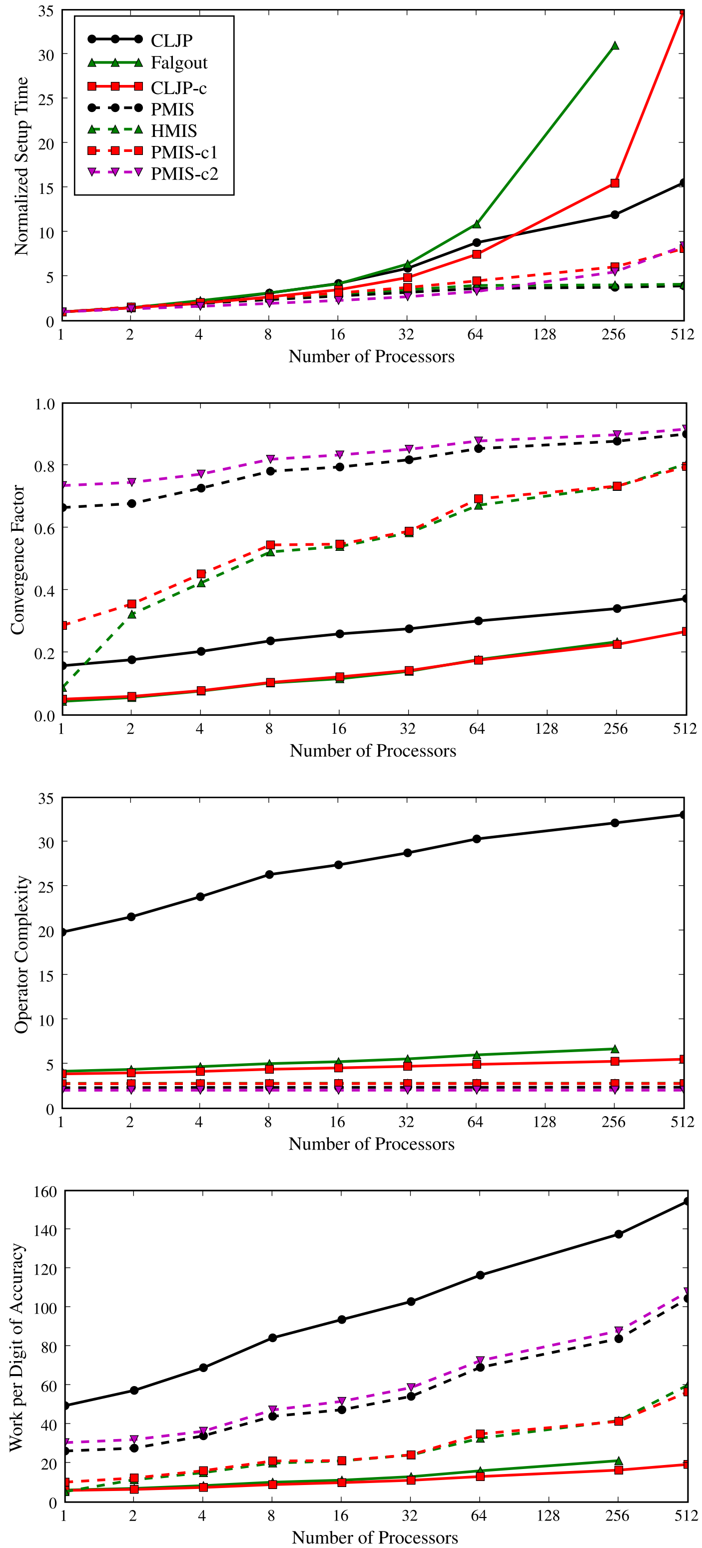


Figure 4.14: Results for the scaled 7-point Laplacian problem. The legend from the first plot applies to all four plots.

Neither CLJP-c nor Falgout are producing large operator complexities, so the extra cost for these algorithms is due to other operations. A portion of the setup time is due to implementation and data structure issues, and another contribution is from the RS portion of Falgout and the coloring in CLJP-c. To this end, Chapter 5 introduces a new algorithm aimed at improving the efficiency of CLJP-c.

The convergence factors grow for all problem sizes and coarsening algorithms. At 512 processors the convergence factors are growing at approximately the same rate as at two processors. Notice the PMIS-like algorithms (PMIS, HMIS, PMIS-c1, and PMIS-c2) are the slowest to converge. In the case of PMIS and PMIS-c2, the sparsity of the coarse grids selected and also the lack of preservation of the structure of the grid by PMIS lead to the slow convergence factors. Both methods produce coarse grids for which a good interpolation operator exists. The slow convergence observed implies that the prolongation operator is inadequate to compensate for the sparse coarse grids. CLJP-c and Falgout yield the fastest convergence factors since these methods produce coarse grids that work well for the given structured problems.

The PMIS-like algorithms all produce grid hierarchies with much lower operator complexities than other methods, and the operator complexities display little or no growth as the problem size is increased. The performance of CLJP is degraded since the problem is discretized on a logically rectangular grid. The growth of operator complexities produced by CLJP is much larger than that of the other methods, as illustrated in Figure 4.15.

The amount of work per digit-of-accuracy grows since all tests result in growing convergence factors. Despite producing relatively large operator complexities, CLJP-c and Falgout create much cheaper AMG methods for the structured problem than the other methods since the convergence factors are much lower than with PMIS-like methods and the operator complexities are much lower than with CLJP.

3D Unstructured Laplacian

In this section, results are reported for the 3D unstructured Laplacian problem (4.5). The problem on a single processor contains approximately 211,000 unknowns. The largest problem is on 512 processors with approximately 100 million unknowns, which gives an average of 198,000 unknowns per processor. The partition size data for this problem is shown in Figure 4.16. The partition sizes fluctuate and are reflected in the results, especially in the operator complexity plot. Normalized setup times, convergence factors, operator complexities, and work per digit-of-accuracy are reported

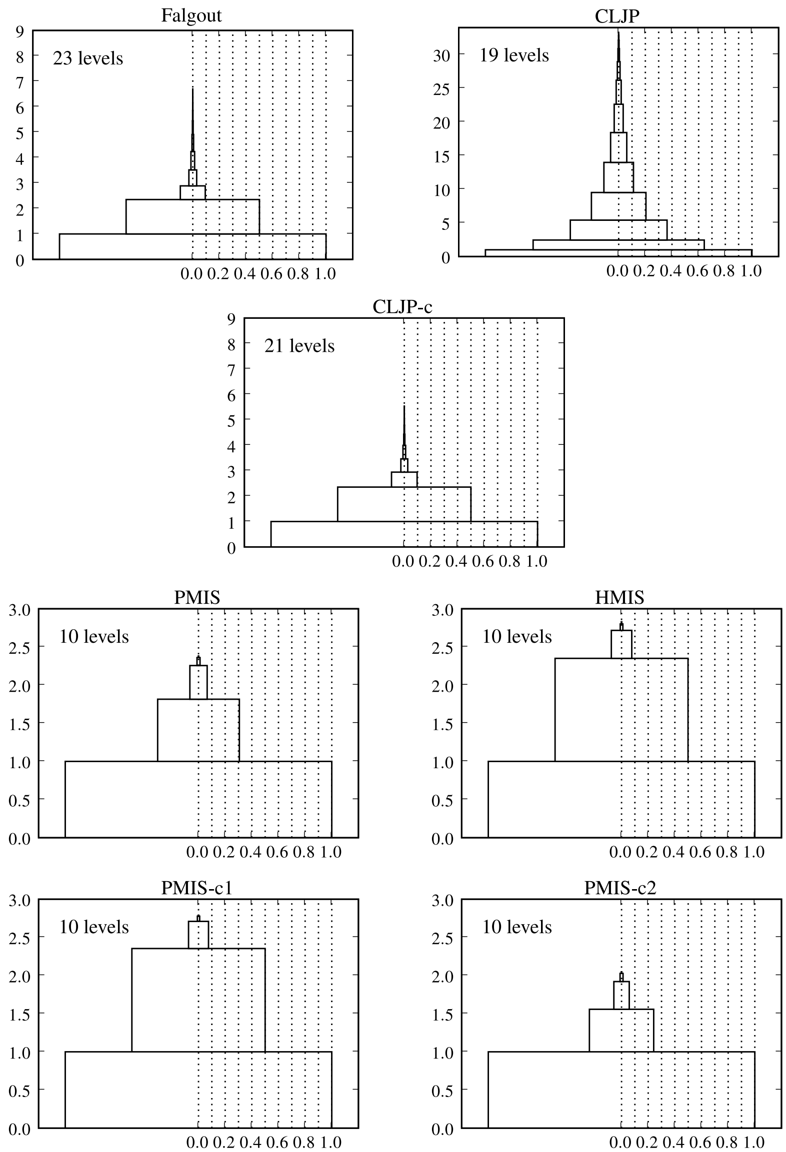


Figure 4.15: Tower plots for the 7-point Laplacian scaled problem. The towers shown are for the 512 processor trials. Notice the scale is not the same in each plot.

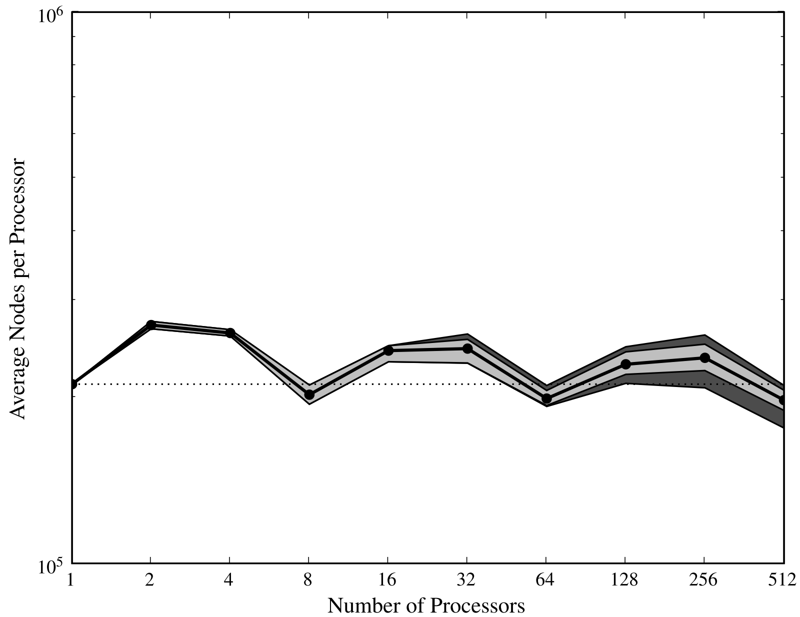


Figure 4.16: Partition size data for the 3D unstructured Laplacian scaled problem and the 3D anisotropic scaled problem.

in Figure 4.17.

As in the structured problem, the setup times for the RS-like algorithms are observed to be growing as the problem size grows. A twenty-fold increase in setup time from one processor to 512 processors is observed. As before, the PMIS-like algorithm setup times are growing, but at a much slower rate than the RS-like algorithms.

The convergence factors are similar to the structured case with one major difference: both CLJP and PMIS perform better on unstructured meshes than on structured meshes. In the structured problem, several groups of lines were present in the plot. However, two groups now appear in the plot: one for the RS-like algorithms and one for the PMIS-like algorithms, meaning CLJP and PMIS both perform as well as algorithms related to them. Also, convergence was slower than for the structured problem, and in all cases the convergence factors increased as the problem size increased.

The operator complexity results demonstrate that PMIS-like methods produce grid hierarchies with extremely low operator complexities which do not grow as the problem size grows. There is little variation in the operator complexities produced by each of the PMIS-like algorithms and little variation is apparent in the tower plots in Figure 4.18. The other algorithms produce operator complexities that are much larger and increase as the problem size grows. Moreover, Falgout coarsening

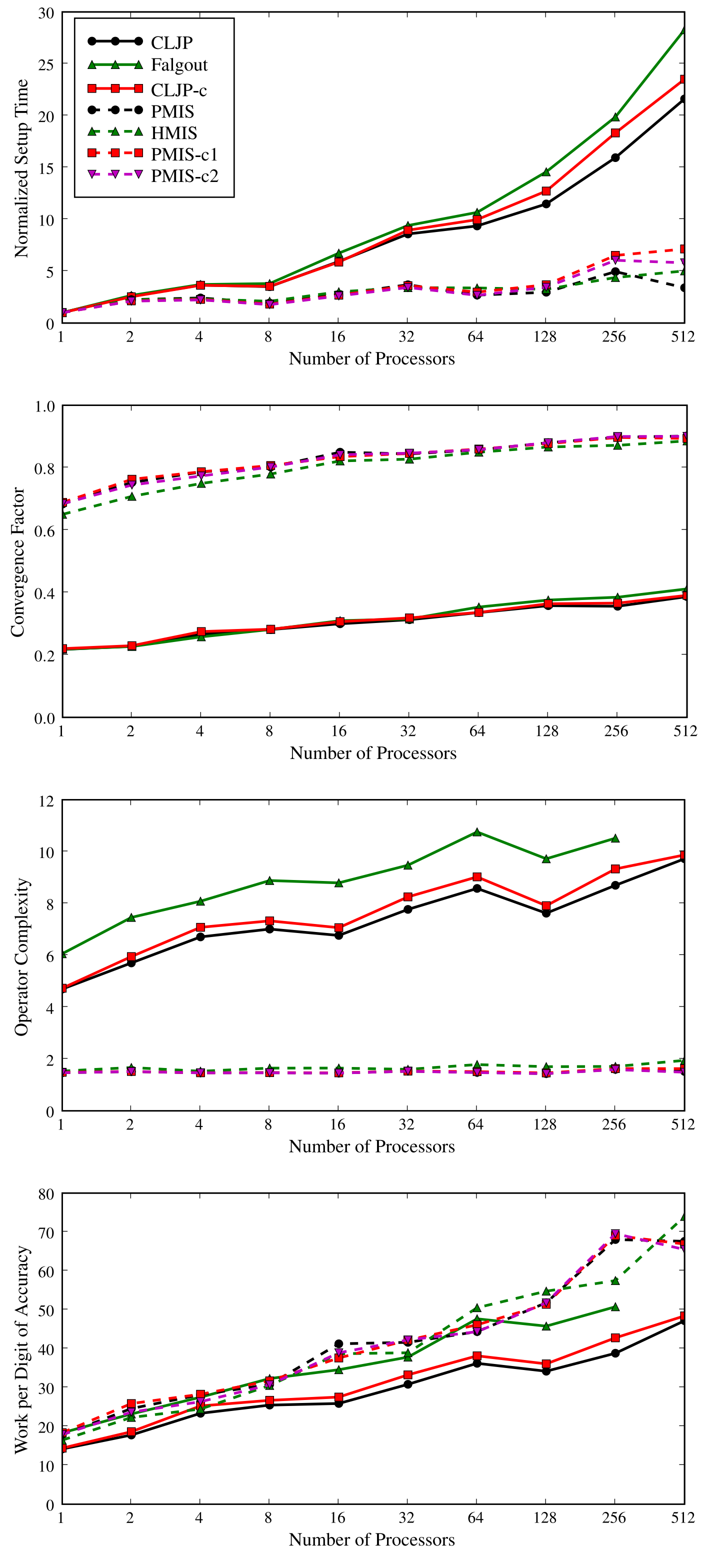


Figure 4.17: Results for the 3D unstructured Laplacian scaled problem. The legend from the first plot applies to all four plots. The final data point for the Falgout line was removed from the final two plots because the operator complexity data was corrupted by overflow.

produces operator complexities consistently and significantly larger than those produced by CLJP and CLJP-c.

The work per digit-of-accuracy results show CLJP to be the cheapest method available for the unstructured case, with CLJP-c a close second. The large difference in CLJP performance on structured versus unstructured grids is again highlighted. PMIS-c2 is the most expensive method, but is comparable to the other PMIS-like algorithms.

3D Unstructured Anisotropic Problem

A 3D unstructured anisotropic problem is defined as follows:

$$\begin{aligned} -(0.01u_{xx} + u_{yy} + 0.0001u_{zz}) &= 0 \quad \text{on } \Omega \qquad (\Omega = (0,1)^3), \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{4.6}$$

The sizes for (4.6) are identical to those in the 3D unstructured Laplacian from the previous section. On one processor, the problem has approximately approximately 211,000 unknowns. On 512 processors there is approximately 100 million unknowns, giving an average of 198,000 unknowns per processor. Figure 4.19 plots the observed normalized setup times, convergence factors, operator complexities, and work per digit-of-accuracy for this experiment. The effects of the non-uniform partitioning are now evident. Comparing the pattern of growth in setup time in Figure 4.19 with the partition data in Figure 4.16, the fluctuations in work per processor affect both setup time and operator complexity.

The normalized setup time results for (4.6) are similar to the results from the 3D unstructured Laplacian setup time data (Figure 4.17). The rate of setup time growth, however, is lower in this problem compared to the isotropic problem, while the convergence factors are higher than in any other problems tested. In each case, the convergence factors approach one. Operator complexities for the anisotropic problem are similar to, but slightly smaller than, the complexities observed in the isotropic problem. The tower plots in Figure 4.20 show the complexities on each level in more detail. Finally, the amount of work needed for one more digit-of-accuracy in the residual is large compared to all other problems examined, which is due to the slow convergence observed.

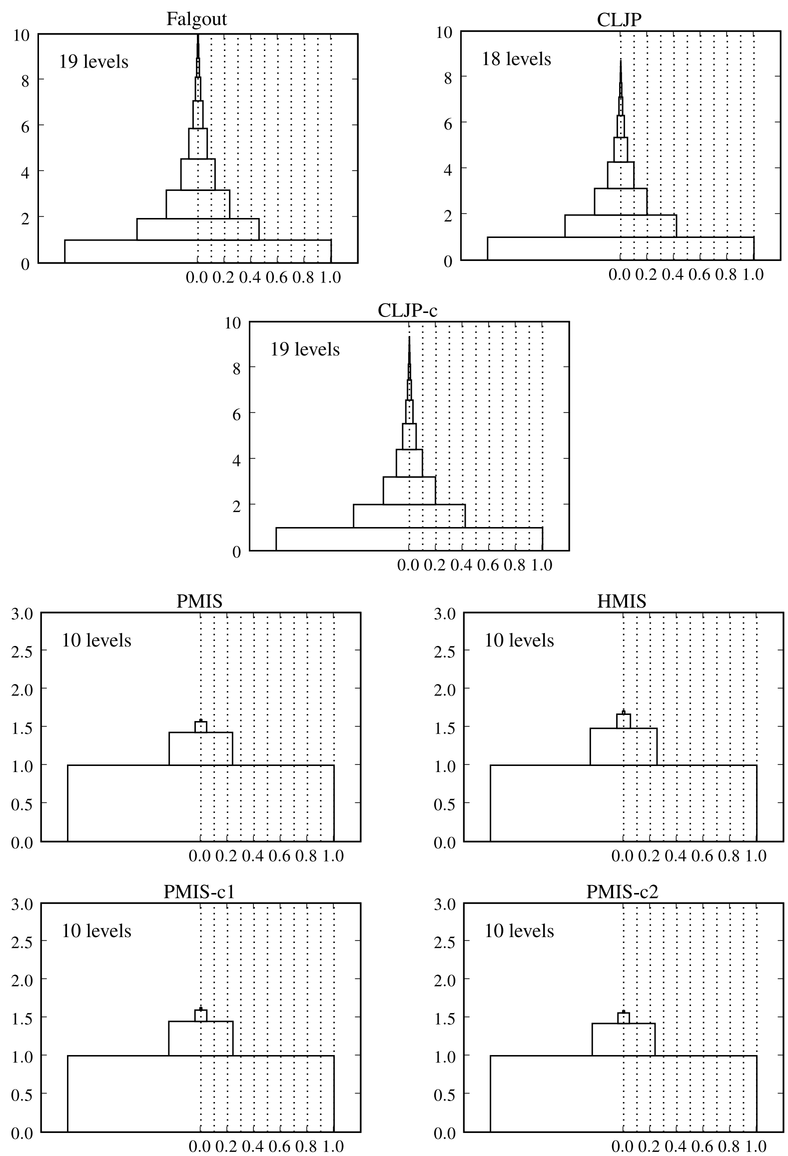


Figure 4.18: Tower plots for the 3D unstructured Laplacian scaled problem. The towers shown are for the 256 processor trials.

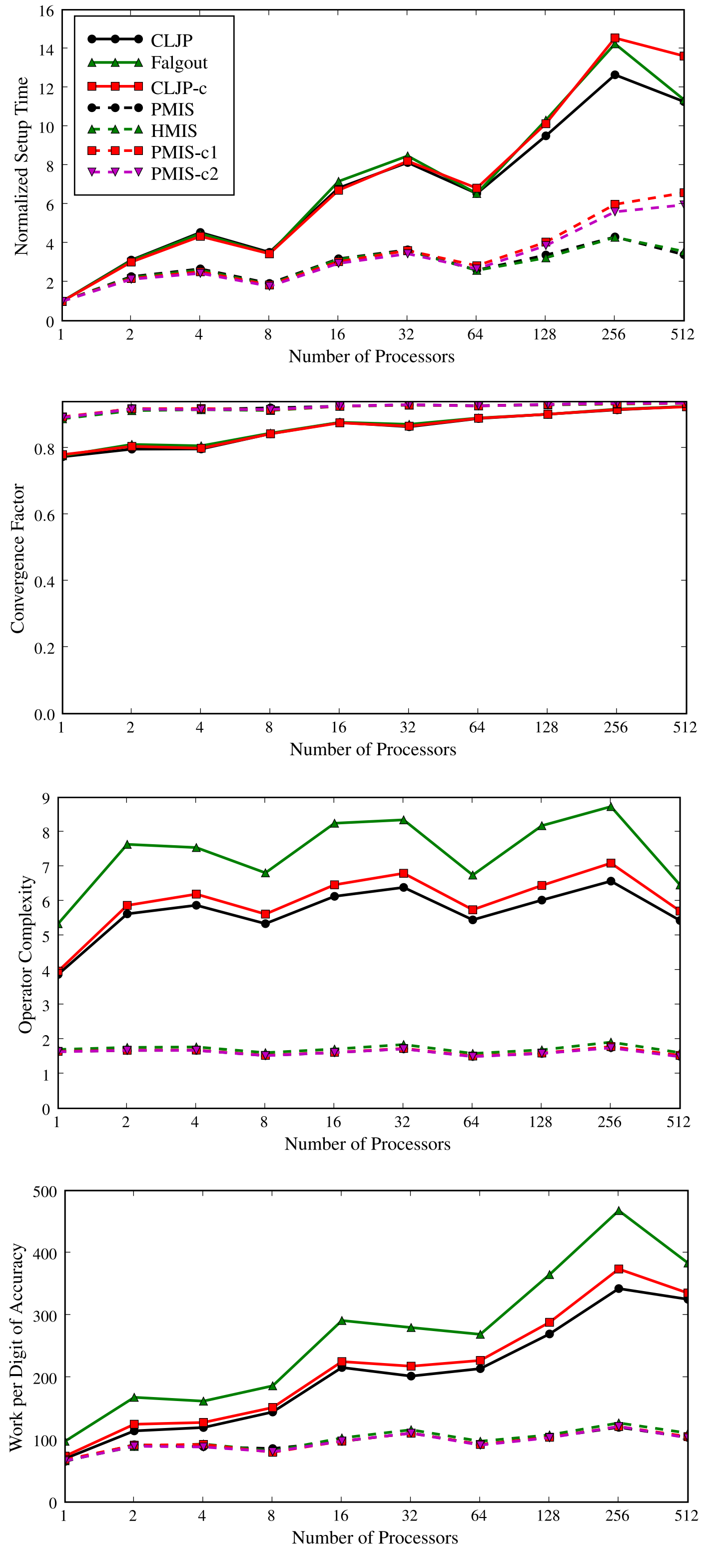


Figure 4.19: Results for the 3D unstructured anisotropic problem. The legend from the first plot applies to all four plots.

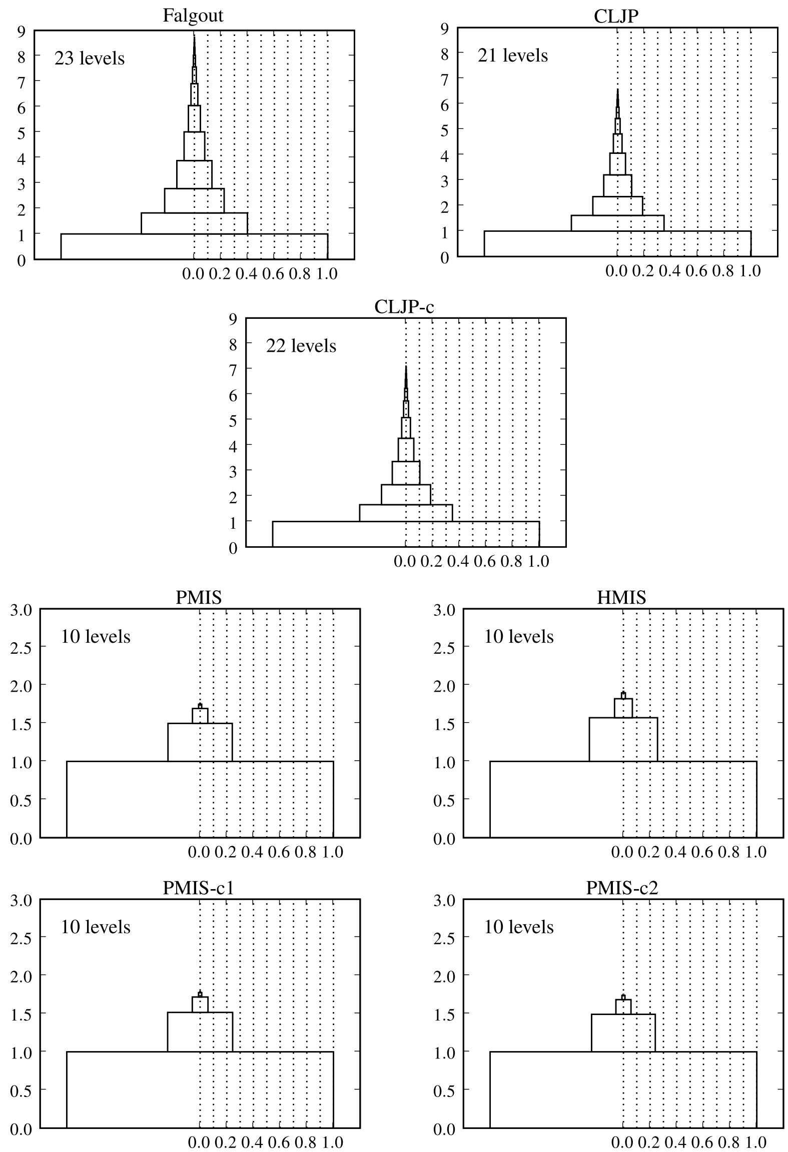


Figure 4.20: Tower plots for the 3D unstructured anisotropic problem. The towers shown are for the 256 processor trials.

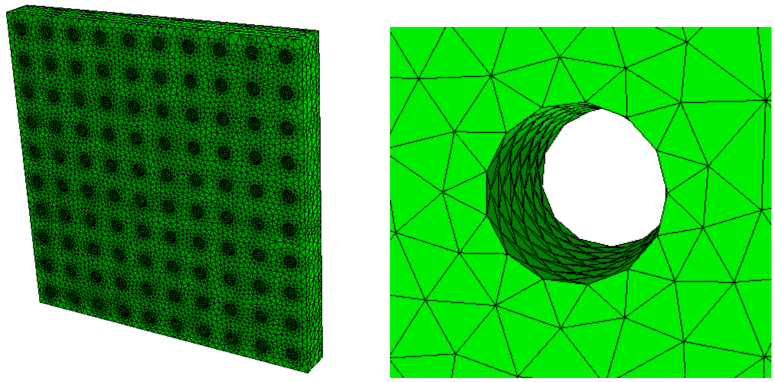


Figure 4.21: The problem domain for the 3D Laplacian holes test problem. The right image is a close-up view of one of the holes.

3D Laplacian Holes

The purpose of this experiment is to examine the effect on the performance of coarsening algorithms on a problem with a more complicated geometry. A thin slab with many holes drilled completely through the material is used as the problem geometry (see Figure 4.21), creating more boundaries than earlier problems.

The problem solved on this domain is once again the Laplacian:

$$\begin{aligned} -\Delta u &= 0 \quad \text{on } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{4.7}$$

On one processor the problem receives approximately 380,000 unknowns. On 512 processors the problem has about 167 million unknowns, giving an average of 327,000 unknowns per processor. Figure 4.22 plots the normalized setup time, convergence factor, operator complexity, and work per digit-of-accuracy data from these tests. The normalized setup time results are similar to Figure 4.17, except the growth in time is less pronounced for (4.7). Falgout, CLJP, and CLJP-c experience the greatest increase in setup times.

The convergence factors for (4.7) are initially lower for each coarsening algorithm than in the unstructured Laplacian problem, but are similar for larger problems. This is due to the large increase of interior vertices relative to the boundary vertices for the largest trials.

Operator complexities in the 3D unstructured Laplacian on the unit cube versus on the holes geometry (4.7) are also similar. Operator complexities are lower in this test, but the rates of growth and the performance of the algorithms relative to one another are similar. The tower plots

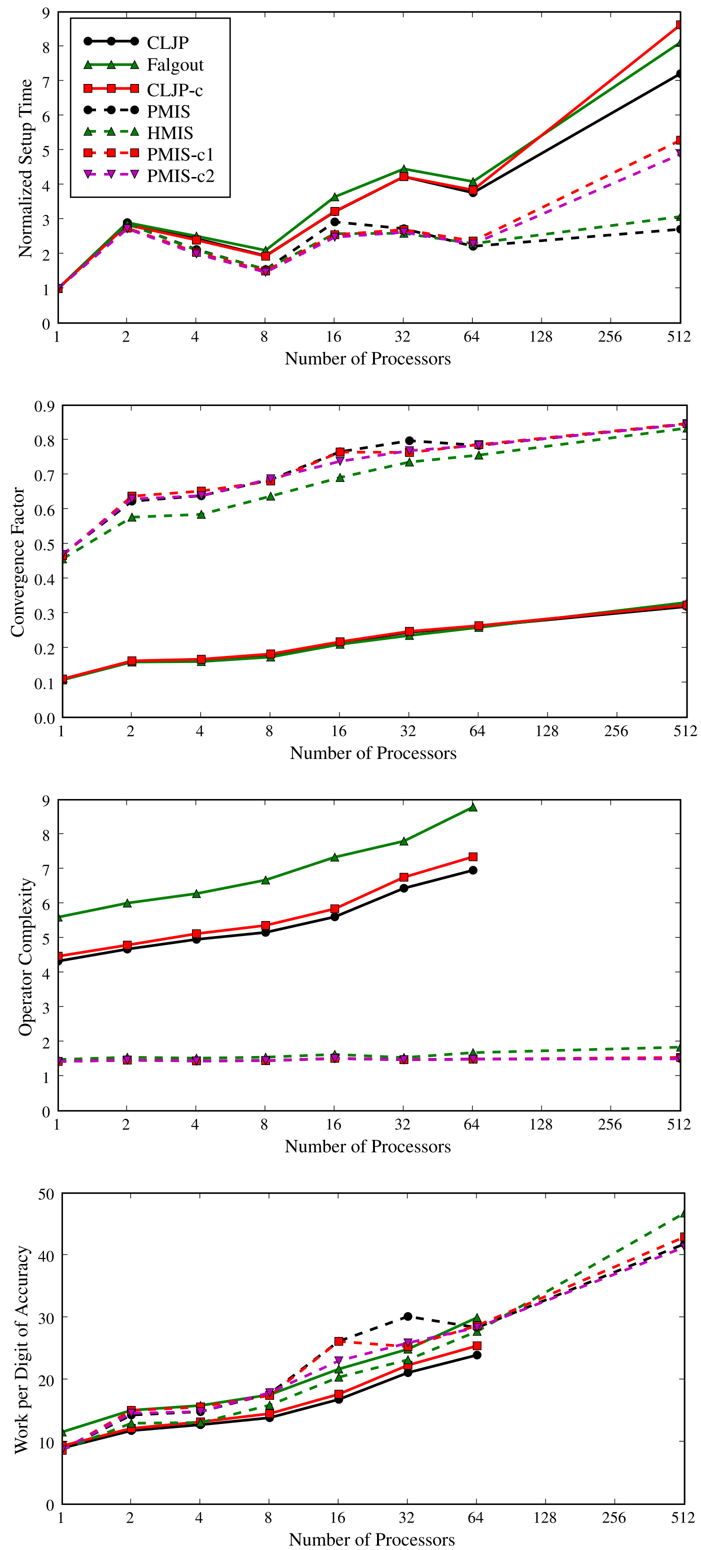


Figure 4.22: Results for the 3D unstructured Laplacian problem on the holes geometry. The legend from the first plot applies to all four plots. The final data points have been removed from several of the lines on the operator complexity and work per digit-of-accuracy plots due to overflow.

in Figure 4.23 show significant differences compared to the tower plots for the 3D unstructured Laplacian (Figure 4.18).

With the complexities and convergence factors behaving similarly between the 3D unstructured Laplacian on the unit cube versus on the holes geometry, the work per digit-of-accuracy results are also similar. A clear difference is that (4.7) is less expensive to solve than the unstructured Laplacian on the unit cube due to slightly lower convergence factors and lower operator complexities.

Between this problem and the unstructured Laplacian on the unit cube, the most noticeable difference is that (4.7) is less expensive to solve and exhibits less growth in setup time. Overall, creating a larger "surface area" yields a geometry that is easier for coarsening algorithms to operate on, but the general characteristics of the solver's performance do not change significantly.

4.6 Conclusions

In this chapter, a new technique is introduced which is designed to address large operator complexities in grid hierarchies generated by CLJP for structured two-dimensional and three-dimensional problems. Modifications to the weight initialization step in CLJP to include information related to the structure of the problem domain leads to improved performance and lower operator complexities. The structural information is computed by graph coloring algorithms. Color weights (i.e., weights unique to each color in the graph) are included in the weight initialization of CLJP to encourage preferential selection of certain sets of vertices. The coloring technique and related modifications produce the CLJP-c algorithm.

The experiments in this chapter demonstrate CLJP-c consistently produces lower operator complexities and smaller solve times compared to CLJP. Furthermore, CLJP-c's performance is often similar to, or better than, Falgout coarsening.

Application of CLJP-c to PMIS leads to the development of two color-based H1′ coarsening algorithms: PMIS-c1 and PMIS-c2. PMIS-c1 results from a direct application of the coloring idea in CLJP-c. A more aggressive approach is taken in the development of PMIS-c2 by producing coarse grids with C-points as distant from one another as allowed by heuristic H1′.

A series of experiments examine the behavior of coarsening algorithms under different conditions. Run time, convergence factors, operator complexities, and work per digit-of-accuracy are reported in each test, revealing unique behavior. In general, PMIS-like algorithms always produce grid hierarchies with lower operator complexities, and RS-like algorithms usually yield methods with

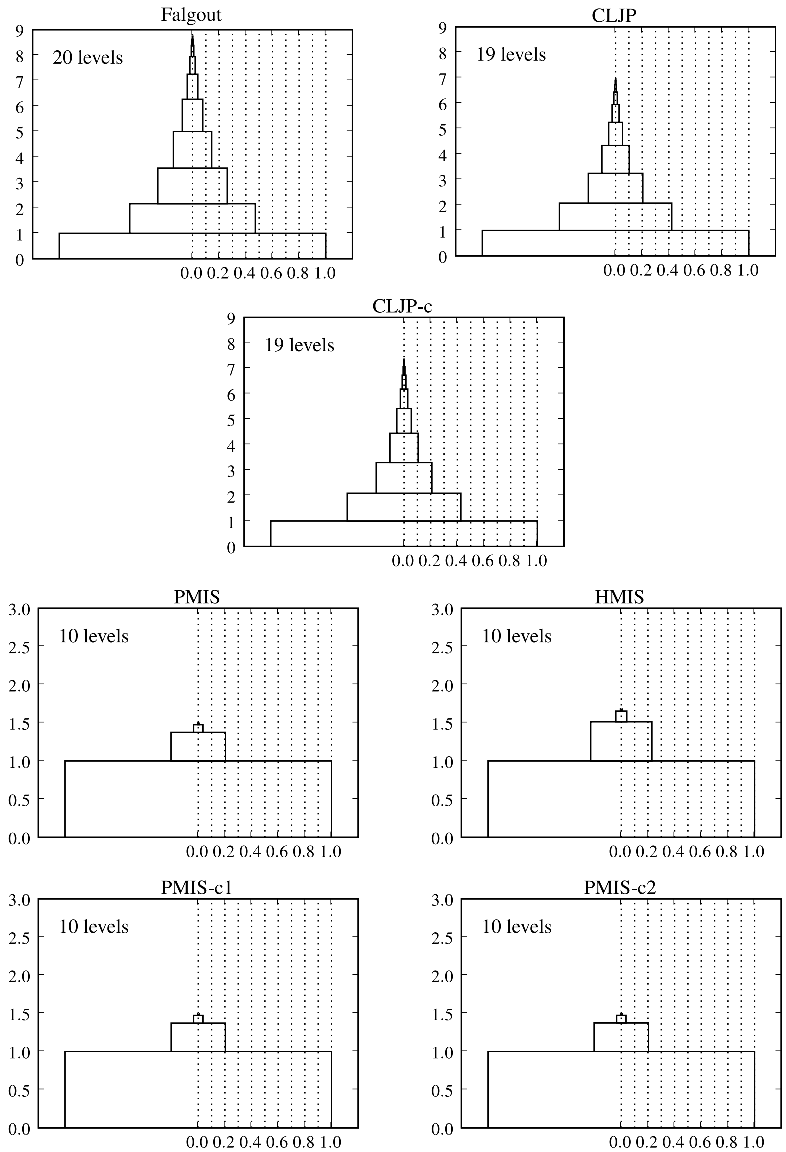


Figure 4.23: Tower plots for the 3D unstructured Laplacian problem on the holes geometry. The towers shown are for the 64 processor trials.

smaller convergence factors. In some tests, such as the anisotropic diffusion problem in Section 4.5.3, AMG convergence is prohibitively slow.

Chapter 5

Bucket Sorted Independent Sets

5.1 Introduction

Chapters 3 and 4 demonstrate the behavior of independent set-based coarse-grid selection algorithms. One common design property among the selection algorithms is a routine to search for vertices to become C-points. Additionally, an update in the weights of vertices as the coarse-grid selection often follows. In this chapter, new theory and algorithms to decrease the computational cost associated with the search and weight update procedures used in coarse-grid selection are presented.

5.2 CLJP-c

Recall that CLJP-c colors the graph of S before selecting C-points, and the colors are used as one component of the vertex weights. As a result, the structure of the coarse grids selected is improved. More formally, the use of graph coloring provides the following important result.

Theorem 5.2.1. *For all pairs of vertices i and $j \in \mathcal{N}_i$ CLJP-c guarantees $w_i \neq w_j$.*

Proof. Assume two adjacent vertices i and j have the same weight. That is, $|S_i^T| = |S_j^T|$ and the weight augmentation provided through coloring is the same for i and j. The graph of S, however, is colored such that i and j are different colors for all $j \in \mathcal{N}_i$, so a contradiction is reached. □

Theorem 5.2.1 establishes that all adjacent vertices have different weights in CLJP-c, which is not guaranteed in CLJP, although is unlikely to occur. The following corollaries are a result of Theorem 5.2.1.

Corollary 5.2.1. *Any set of vertices in the graph of S that share the same weight is an independent set.*

Corollary 5.2.2. *The set of vertices with the largest weight in the graph of S form an independent set satisfying 5.1. That is, each vertex in that independent set has a uniquely maximal weight in its* *neighborhood.*

The first corollary states that independent sets can be selected in the graph simply by selecting sets of vertices with the same weight. Corollary 5.2.2 refines this observation to a subset of vertices guaranteed to satisfy the selection criterion. In particular, it shows it is possible to build the coarse grid by selecting vertices with the maximum weight, updating weights, selecting the next set of vertices with maximum weight, and so on. This approach is taken by the algorithm developed in Section 5.4. It is proven in Section 5.4.1 that despite the difference in approach, the resulting coarse grid is the same as that selected by CLJP-c.

5.3 Coarse-Grid Selection Search and Weight Update

CLJP and its descendents select an independent set D in each iteration. Stated originally in Section 3.4.2, the condition for selecting D is that all vertices $i \in D$ must satisfy

$$w_i > w_j \text{ for all } j \in \mathcal{N}_i. \tag{5.1}$$

CLJP and CLJP-c rely on a search routine to locate locally maximal weights and on a weight update routine to modify weights of vertices connected to new C-points.

The algorithms for searching and updating vertex weights in CLJP in detail are examined in this section. In particular, the impact of using a sparse matrix format on the coarsening procedure. The pseudo-code below assumes the software uses a compressed sparse row (CSR) [49] matrix format or other similar format, which are common matrix formats in numerical software. CSR provides low memory costs for storing sparse matrices and provides efficient access to the nonzeros in a row. Accessing the nonzeros in a column is an expensive operation in this format and strongly influences the weight update routine in CLJP-style algorithms because S is, in general, not symmetric.

The search step in CLJP is implemented as shown in Algorithm 5.1. In the first iteration, Line 3 is run $2|E|$ times. The total cost of search in constructing the coarse grid depends on the number of iterations needed. Even in the best case of $\Omega(E)$ time, the cost is significant when the graph contains large numbers of edges, as usually happens on the lower levels in the grid hierarchy (see [3] for examples). In the next section, a new technique is introduced for conducting the search in coarse-grid selection algorithms independent of the number of edges in the graph.

Pseudo-code for the weight update in CLJP is shown in Algorithm 5.2. The level of complication in this update routine is due to the CSR format and the need to find vertices strongly influenced

Algorithm 5.1 Coarse-Grid Selection Graph Search

Search-Graph(S, C, F) {

$D \leftarrow \emptyset$
for all $i \notin (C \cup F)$ **do**
  if $w_i > w_j, \forall j \in \mathcal{N}_i$ **then**
    $D \leftarrow D \cup \{i\}$
  end if
end for
return D

}

by new C-points. When a new C-point k is selected, the first type of weight update is trivial since determining the vertices in S_k is inexpensive. The second type of update is more expensive since the vertices influenced by k are difficult to determine in a CSR format. The update requires a search of many vertices i and all of their strong influencing neighbors j. The routine then searches strongly influencing j to determine if any $k \in D$ strongly influences both i and j. The cost increases dramatically as the density of S increases. Large operator complexities have a disproportionately large impact on coarse-grid selection run time. In the next section, a modified update routine to compliment the new search technique is introduced.

5.4 Bucket Sorted Independent Set Selection

In this section, new techniques for searching the graph of S for new C-points and subsequently updating the weights of remaining vertices are developed. The new algorithm is labeled Bucket Sorted Independent Sets (BSIS) to reflect the data structure used.

Like CLJP-c, BSIS depends on graph coloring, but utilizes modified routines for search and weight update. Furthermore, rather than applying the color information to augment vertex weights, BSIS uses the colors in a bucket data structure. Once initialized, this data structure selects independent sets, which satisfy the conditions in (5.1), in constant time.

5.4.1 Coarse Grid Invariance

Theory is developed in this section to prove coarse-grid selection invariance in general independent set-based algorithms. The algorithms considered thus far select independent sets using (5.1), meaning i is eligible to be in D if its weight is larger than the weights of vertices in its neighborhood $\mathcal{N}_i$. Recall the neighborhood of i (Definition 3.3.1) is the set of vertices strongly influenced by i or strongly influencing i. Algorithms relying on different and larger neighborhoods, such as *distance-d*

Algorithm 5.2 CLJP Weight Update for CSR Matrix

```
UPDATE-WEIGHTS(S, D, C, F, w) {
for all d ∈ D do
   for all i ∈ S_d do
      w_i ← w_i − 1  /* see Figure 3.5 left */
      S_d = S_d \ {i}  /* removing edge from graph */
   end for
end for
for all i ∉ (C ∪ F) do
   for all k originally in S_i do
      if k ∈ D then
         mark k NEW-C-POINT
      end if
   end for
   for all j ∈ S_i do
      if j ∉ D then
         for all k ∈ S_j do
            if k is marked NEW-C-POINT then  /* i and j mutually influenced by k */
               w_j ← w_j − 1  /* see Figure 3.5 right */
               S_i = S_i \ {j}  /* remove edge from j to i */
            end if
         end for
      end if
   end for
   for all k originally in S_i do
      if k ∈ D then
         unmark k
         S_i = S_i \ {k}  /* remove edge from k to i, if present */
      end if
   end for
end for
}
```

neighborhoods, are conceivable.

Definition 5.4.1. *The* distance-d neighborhood *of* i*, denoted* $\mathcal{N}_i^d$*, is the set of vertices within* d *hops of* i *in the symmetrized strength matrix, excluding* i*. That is,* $\mathcal{N}_i^d = \left[\bigcup_{j\in\mathcal{N}_i^{d-1}} (\mathcal{N}_j \cup \{j\}) \cup \mathcal{N}_i\right] \setminus \{i\}$*, where* $d > 0$ *and* $\mathcal{N}_i^0 = \emptyset$*.*

For a general independent set-based coarse-grid selection algorithm, a vertex i is eligible to be added to D if

$$w_i > w_j \text{ for all } j \in \mathcal{N}_i^s, \tag{5.2}$$

where $\mathcal{N}_i^s$ is the *selection neighborhood* of i. Although the distance-d neighborhood is a sensible choice for the selection neighborhood, this discussion is not limited to such cases. It is assumed, however, the matrix formed by $\mathcal{N}_*^s$ (i.e., the selection sets for all vertices) is symmetric. This is equivalent to stating if $j \in \mathcal{N}_i^s$, then $i \in \mathcal{N}_j^s$ for all i and j in the vertex set. Furthermore, this discussion assumes the weight of vertex i is only decremented following weight updates and is never modified due to the assignment of a vertex $j \notin \mathcal{N}_i^s$ to the C-point set.

Given a symmetric, but otherwise arbitrary, set of selection neighborhoods, the set of vertices potentially able to affect the vertex weight of i or $j \in \mathcal{N}_i^s$ is the *extended selection neighborhood*.

Definition 5.4.2. *The* extended selection neighborhood *of i, denoted $\mathcal{N}_i^{2s}$, is the union of the selection neighborhood of i with the selection neighborhoods of the vertices in $\mathcal{N}_i^s$, excluding i. That is, $\mathcal{N}_i^{2s} = \left[\left(\bigcup_{j\in\mathcal{N}_i^s} \mathcal{N}_j^s\right) \cup \mathcal{N}_i^s\right] \setminus \{i\}$.*

When a vertex i satisfies the generalized selection condition (5.2), no other C-point assignments affect w_i. This is formalized below.

Lemma 5.4.1. *If (5.2) is satisfied for vertex i, then i must be the next vertex in $\{i\} \cup \mathcal{N}_i^s$ to become a C-point, regardless of any other selections and updates made in the graph.*

Proof. The satisfaction of (5.2) means $w_i > w_j$ for all $j \in \mathcal{N}_i^s$. For i to not become a C-point, its weight must become smaller than some $j \in \mathcal{N}_i^s$. The weight of i, however, is not decremented unless some $j \in \mathcal{N}_i^s$ satisfies (5.2) and becomes a C-point, which is impossible until after i is assigned to the C-point set. ☐

To demonstrate all algorithms using the same selection neighborhood and update rules select identical coarse grids, a proof with an inductive argument is presented below. The base case is provided by the first set of vertices satisfying the general selection conditions.

Definition 5.4.3. *Let D_0 be the set of all vertices satisfying (5.2) in the first iteration of coarse-grid selection.*

Vertices in D_0 are destined to become C-points regardless of the algorithm used to build coarse grids. In the independent set-based algorithms discussed in Chapters 3 and 4, all D_0 vertices become C-points in the first iteration. Any algorithm constructed, however, eventually selects all D_0 vertices as C-points, as proven in the following lemma.

Lemma 5.4.2. *Given the same selection neighborhoods and update heuristics, all algorithms select vertices in D_0 as C-points.*

Proof. The proof follows from Lemma 5.4.1. All vertices in D_0 satisfy the selection condition (5.2), so the assignment of any other C-point has no effect on the weight of a D_0 vertex. ☐

Lemma 5.4.2 states that any coarse-grid selection method using the general selection condition invariably selects D_0 vertices as C-points. This result is used in the proof of the next theorem.

Theorem 5.4.1. *All independent set-based coarse-grid selection algorithms given the same initial weights and using the same selection neighborhood, selection criterion based on (5.2), and weight update heuristics as described above select identical coarse grids.*

Proof. Let c be the vertices in $\mathcal{N}_i^{2s}$ satisfying the conditions for D in some arbitrary iteration. Suppose assigning vertices $c_1 \subset c$ to the C-point set leads to $w_i > w_j$ for all $j \in \mathcal{N}_i^s$. Also suppose assigning $c_2 \subset c$, $c_1 \neq c_2$, to the C-point set leads to the existence of some $j \in \mathcal{N}_i^s$ such that $w_j > w_k$ for all $k \in \mathcal{N}_j^s$.

For both conditions to be true, one or both of the following cases must be satisfied.

1. The value of w_i is smaller when c_2 is added to the C-point set than when c_1 is added. For this case, it must be true that $|c_2 \cap \mathcal{N}_i^s| > |c_1 \cap \mathcal{N}_i^s|$.

2. The value of w_j is larger when c_2 is added to the C-point set than when c_1 is added. For this case, it must be true that $|c_2 \cap \mathcal{N}_j^s| < |c_1 \cap \mathcal{N}_j^s|$.

Case 1 creates a contradiction. If $|c_2 \cap \mathcal{N}_i^s| > |c_1 \cap \mathcal{N}_i^s|$, then $(c \setminus c_1) \cap \mathcal{N}_i^s \neq \emptyset$. Following the assignment of the vertices in c_1 to the C-point set, there remains some $k \in \mathcal{N}_i^s$ that is also in c. Therefore, $w_k > w_i$, contradicting the first assumed condition.

Case 2 is similarly impossible. If $|c_2 \cap \mathcal{N}_j^s| < |c_1 \cap \mathcal{N}_j^s|$, then $(c \setminus c_2) \cap \mathcal{N}_j^s \neq \emptyset$. Following the assignment of the vertices in c_2 to the C-point set, there remains some $k \in \mathcal{N}_j^s$ that is also in c. Therefore, $w_k > w_j$, contradicting the second assumed condition.

Both cases are impossible, so the order of C-point selection within the selection neighborhood of each vertex is invariant. Combined with Lemma 5.4.2 as the base case, this proves by induction the invariance of coarse-grid selection for all algorithms using identical selection conditions. □

Remark 5.4.1. *CLJP and CLJP-c use the distance-one neighborhood as the selection neighborhood, (5.1) as the selection criterion, and the weight update heuristics in Algorithm 3.4. Given the same initial weights, all algorithms based on the parameters utilized by CLJP and CLJP-c select identical coarse grids.*

Theorem 5.4.1 is an important result about the nature of coarse grids selected by general independent set-based algorithms. This information enables the design and implementation of new algorithms that yield identical coarse grids using different and possibly more efficient techniques.

5.4.2 Bucket Sorted Independent Sets Algorithm

The BSIS algorithm creates a bucket data structure enabling a search for new C-points without individually scanning each vertex and associated edges. Figure 5.1 illustrates the bucket data structure. The number of buckets in the data structure is $\max_{i \in S} |S_i^T|$ times the number of colors in the graph. That is, each possible weight in S has its own bucket. The vertices are distributed to the appropriate buckets during the setup of the coarse-grid selection algorithm, where the bucket of a newly placed vertex depends on the number of vertices it strongly influences and its color. For example, Vertex 14 strongly influences six vertices and is black. Therefore, it is placed into the black bucket in the sixth group of buckets. More notably, the vertices in a bucket form an independent set (e.g., vertices 4, 16, and 21).

In each iteration, the non-empty bucket with largest weight forms D (see Corollary 5.2.2). These vertices are assigned to the C-point set and removed from the data structure. Vertex weight updates lead to corresponding updates to the data structure, and new F-points are removed from the data structure. These operations continue until all buckets are empty, at which point the coarse-grid selection is complete. Algorithms 5.3, 5.4, and 5.5 outline the operations discussed above.

Algorithm 5.3 BSIS Data Structure Setup

```
BSIS-SETUP(S) {
  for all i ∈ V do
    bucketID ← (w_i − 1) · numColors + color_i
    bucket[bucketID].INSERT(i)
  end for
}
```

Algorithm 5.4 Independent Set Selection

```
BSIS-INDEPENDENT-SET-SELECTION(S) {
  return non-empty bucket with largest bucketID
}
```

Algorithm 5.5 BSIS Weight Update

```
BSIS-WEIGHT-UPDATE(S) {
  bucketID ← (w_i − 1) · numColors + color_i
  bucket[bucketID].REMOVE(i)
  bucket[bucketID − numColors].INSERT(i)
}
```

Figure 5.2 illustrates the graph and data structure following the first iteration of the algorithm. Vertex 10 has become a C-point and its neighbors weights have been updated. Vertices assigned to

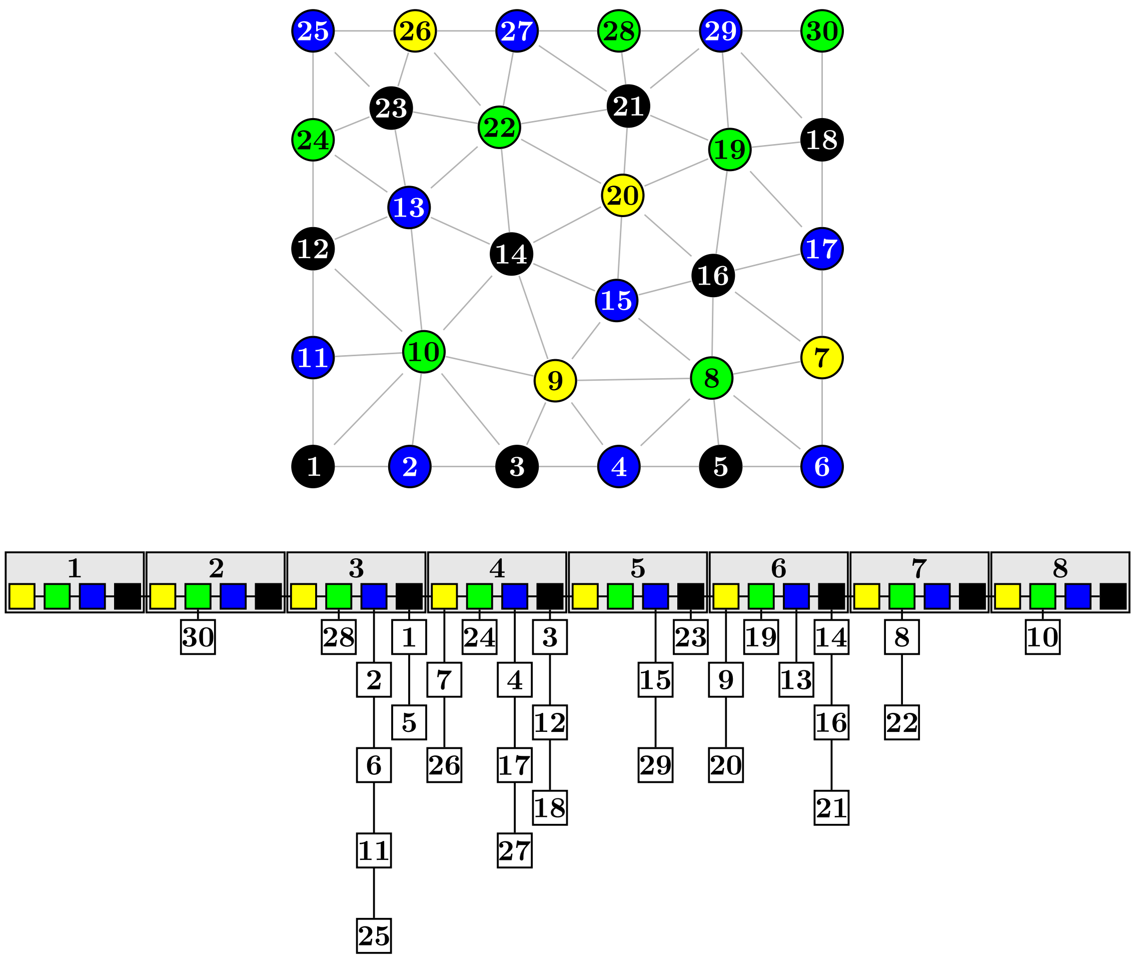


Figure 5.1: The BSIS data structure.

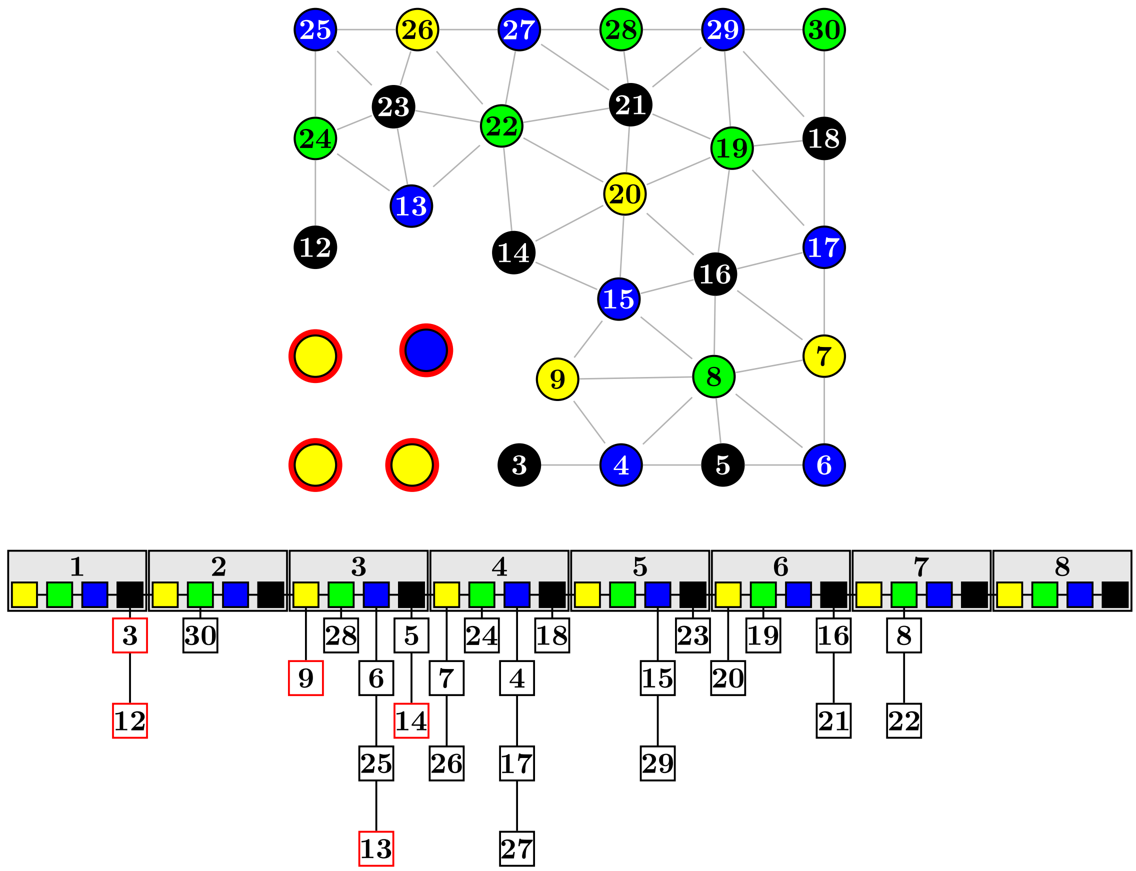


Figure 5.2: The BSIS data structure after selecting the first C-point (Vertex 10). The weights of neighbors of the new C-point are updated. Some neighbors become F-points and are removed from the data structure. Vertices removed from the data structure are highlighted with a red ring in the graph, while other neighbors are moved to new locations in the data structure and are highlighted (to help in reading the figure) in the data structure with a red box.

F or C are removed from the data structure and other affected vertices are moved to new locations in the data structure. Such vertices are highlighted in red.

The weight update routine described in Section 5.3 is very expensive in this context because some iterations of BSIS select few C-points. For a graph with a large number of colors, BSIS may execute dozens or hundreds of low-cost iterations to select a coarse grid. Recall the weight update routine described loops through all unassigned vertices each time it is called, so when it is run by BSIS, work is done on many unaffected vertices, which is computationally inefficient.

The largest factor in the cost of the weight update results from searching for the second type of weight update in Algorithm 5.2, which is done by looping through all unassigned vertices since a new C-point cannot easily determine which vertices it strongly influences in a CSR matrix. It is less expensive in this situation to construct the transpose of S than to search the entire graph in each iteration. In S^T, a C-point quickly determines which vertices it influences and "paints" them. The algorithm then loops through all painted vertices and determines if any are neighbors. This is a simple solution that has a dramatic effect on the performance of BSIS, although the update cost remains approximately equivalent to the cost in CLJP-c. Chapter 7 describes plans for continued investigation into decreasing the cost of weight update in coarsening algorithms.

5.5 Weight Update Aggregation

Whenever a vertex weight is updated, BSIS moves the corresponding vertex to a new location in the data structure. During the selection of the coarse grid, the cost of the updates to the data structure is non-trivial and, as shown in this section, unnecessary.

Only one bucket in the data is touched during the C-point selection step: the largest weight non-empty bucket in the data structure. Other buckets are subsequently affected by the weight updates resulting from new C-point assignments. A different approach is possible, however, since the only bucket that must contain the correct vertices is the one from which C-points are selected.

To save cost, we suggest a lazy approach based on aggregating the cost of the updating vertex weights. Rather than investing computation into maintaining accuracy in the data structure, a less expensive mechanism to test if a vertex is in the correct location is provided. When a weight is updated, the vertex is not moved until it is found in the bucket being used as the new independent set D.

Figure 5.3 depicts the data structure after the first set of C-points is selected. Rather than moving vertices to new buckets, the method now keeps them in the same location and only moves them when necessary. As shown in Section 5.6, aggregation of the weight updates leads to significant savings in computational cost.

5.6 Experimental Results

To demonstrate BSIS, the algorithm is compared with CLJP-c. The test problem is the 3D 7-point Laplacian on a structured grid:

$$\begin{aligned} -\Delta u &= 0 \quad \text{on } \Omega \qquad (\Omega = (0,1)^3), \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{5.3}$$

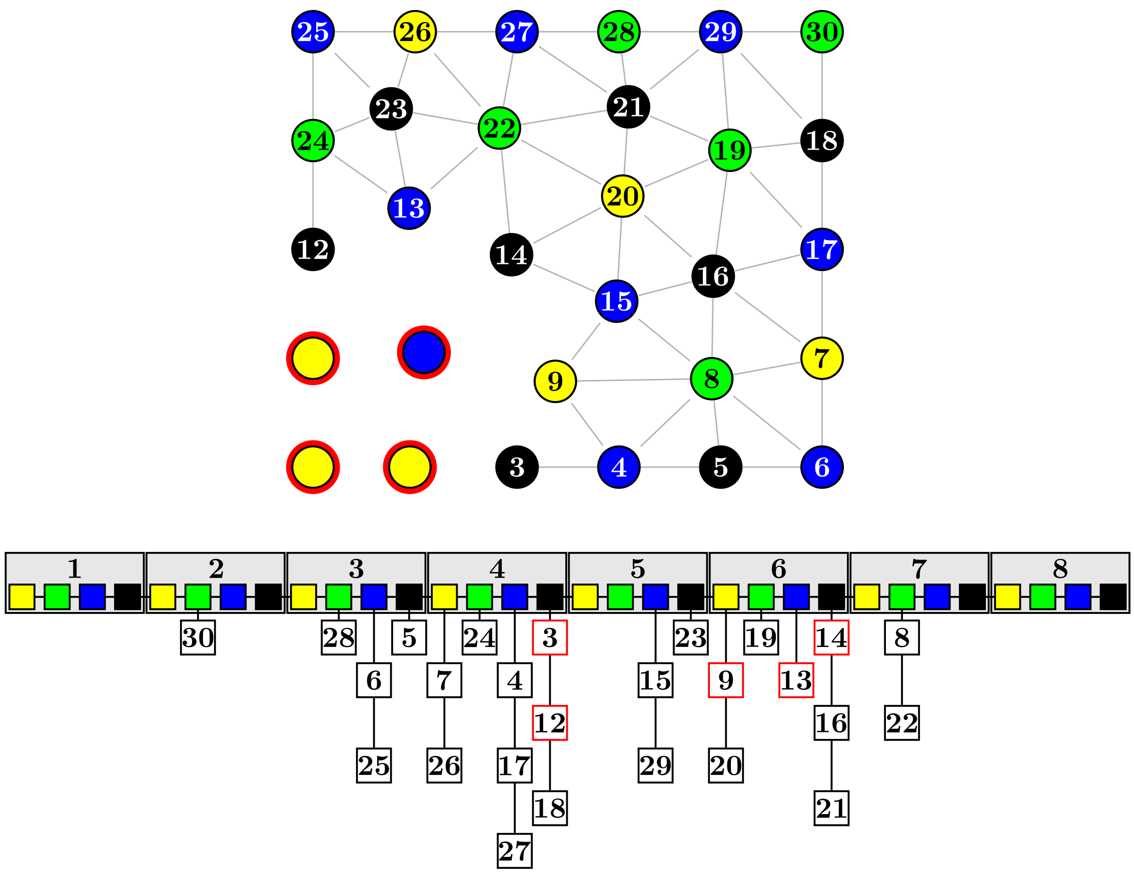


Figure 5.3: The BSIS data structure after selecting the first C-point (Vertex 10) with aggregate weight updates.

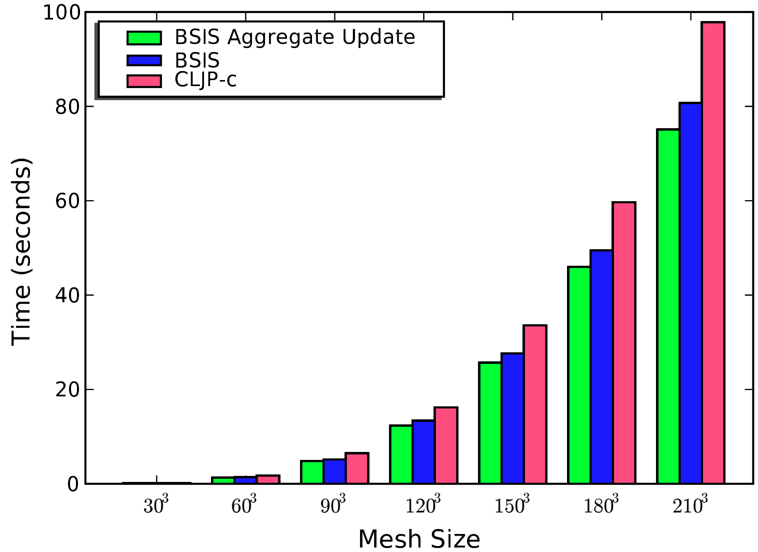


Figure 5.4: Coarse-grid selection times using BSIS with aggregate weight update, standard BSIS, and CLJP-c on the 7-point Laplacian.

A 7-point Laplacian is selected since it is a common initial test problem and structured problems often lead to the largest operator complexities for coarsening algorithms satisfying heuristic H1. By creating larger operator complexities, the algorithms are forced to traverse more edges in the graph, leading to more work.

Timing data for the selection of all coarse grids in the hierarchy is reported. This time includes the cost for all parts of the algorithms, including the graph coloring phase. AMG solve phase information is not reported since the algorithms produce identical coarse grids and since information on solve phase performance for AMG with CLJP-c is documented in [2, 3].

The smallest problem is a $30 \times 30 \times 30$ grid. Subsequent problems are grids of size 60^3, 90^3, up to 210^3. The largest problem is 343 times larger than the smallest problem and contains more than nine million degrees of freedom.

Results for the experiment are presented in Figure 5.4. BSIS completes coarse-grid construction in less time than CLJP-c in every case, and BSIS with aggregate weight update performs significantly better than standard BSIS. For the largest problems BSIS is approximately 17% cheaper than CLJP-c. BSIS with aggregate weight updates is 23% cheaper on the largest problems. The benefit is magnified, relative to CLJP-c, for the smaller problems.

Experiment (5.3) demonstrates the effectiveness and competitiveness of the bucket technique. Increased efficiency for the BSIS algorithm is anticipated through further research. Furthermore, the methods and concepts in this research are also applicable to other coarsening algorithms and possibly to other elements in the AMG setup phase.

5.7 BSIS Variants

The ideas developed in this chapter are applicable to other coarsening algorithms that utilize a graph coloring step, such as color-based methods designed to satisfy heuristic H1′. The PMIS-c1 algorithm is expected to benefit from BSIS and is expected to perform naturally in parallel since vertex weights are not updated in PMIS.

5.8 Parallelizing BSIS

Using BSIS in a parallel algorithm presents challenges because the parallelism in BSIS is very fine-grained. Its elegance and potential to greatly improve the efficiency of coarse-grid selection motivates the development of parallel algorithms incorporating BSIS. Several alternatives are explored in this section.

The first idea is called the boundary painting method and takes advantage of the invariance between coarse grids selected by BSIS and CLJP-c. The idea is to use BSIS to select as many of the interior C-points as possible before doing any communication with neighboring processors. All vertices belonging to a processor are colored and inserted into the BSIS data structure. The processor boundary vertices, however, are "painted". If a painted vertex is in a set D in some iteration, then the vertex is not added to C. It is instead removed from the data structure and its on-processor neighbors are also painted. Figure 5.5 illustrates the first iteration of the painted boundary method with weight update aggregation. The data structure shown is for the left domain. The first iteration selects a new C-point, but does not select any painted vertices. In the second iteration, Vertex 22 is selected, but is already painted. Therefore, on-processor neighbors of Vertex 22 are also painted (see Figure 5.6). The method finishes when the data structure is emptied of all vertices. The product is now three disjoint sets: C and F, as usual, but also a set of painted vertices. The painted vertices are the vertices that cannot be assigned to the F or C set without communication with another processor. The information is provided to CLJP-c, which handles the parallel portion of the algorithm.

The painted boundary approach is ideal given large numbers of interior vertices, which can be guaranteed on most problems for the fine grid. A side-effect of H1-based coarsening algorithms,

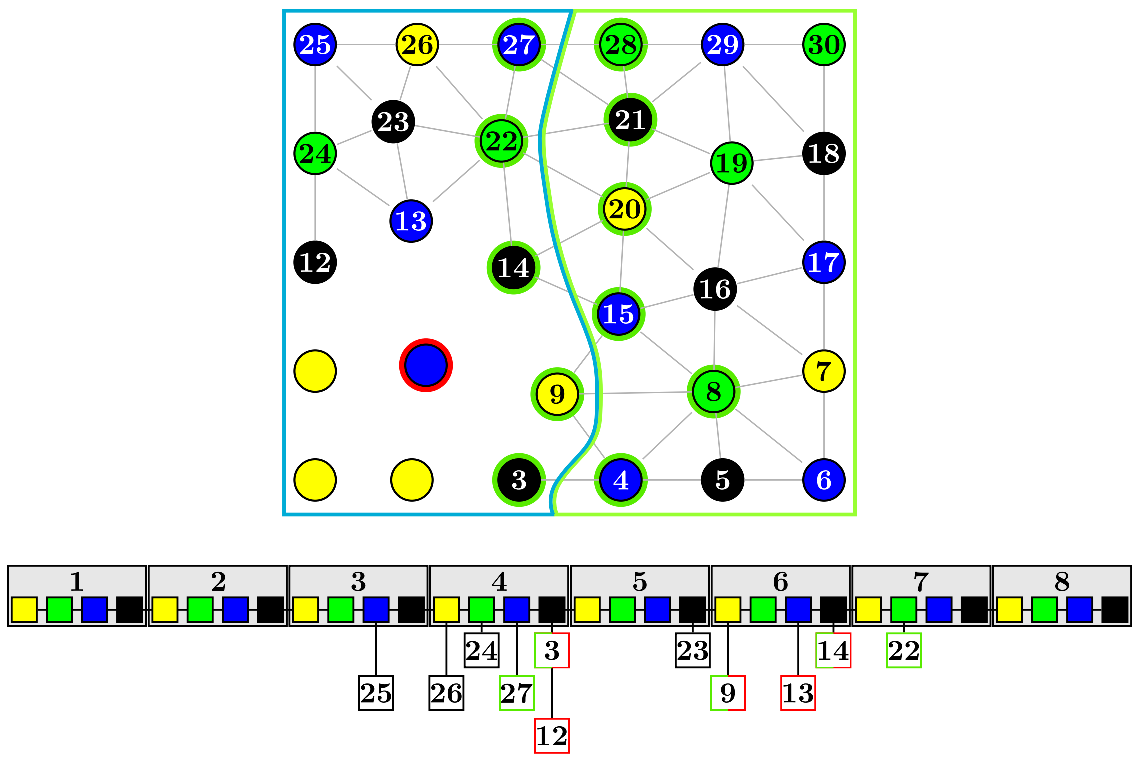


Figure 5.5: The painted boundary method with aggregate weight updates following one iteration. The data structure is for the left domain. Painted vertices are marked with a green ring in the graph and a green box in the data structure. Vertices in the data structure that are half green and half red are painted and also have had their weights updated.

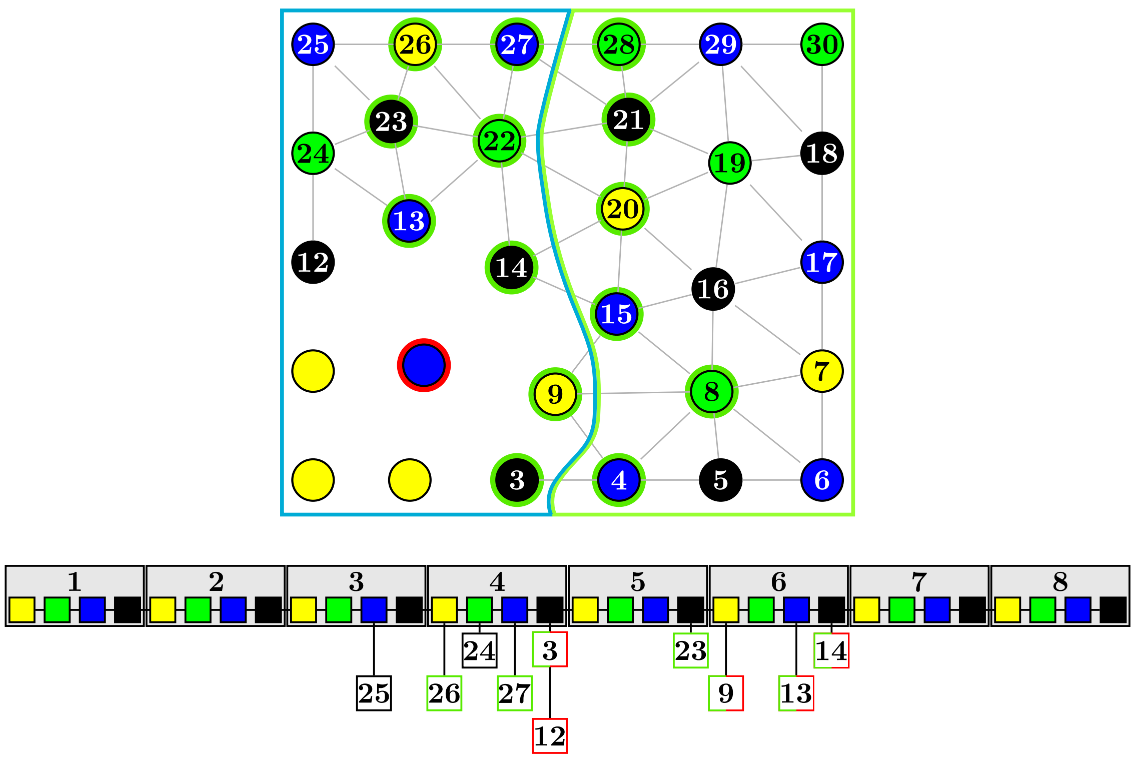


Figure 5.6: The painted boundary method with aggregate weight updates following the second iteration. In this iteration a painted vertex was selected, leading to the painting of its on-processor neighbors.

however, is the creation of denser graphs on coarse levels. One solution is to use the painted boundary method on fine levels and then switch to CLJP-c further along in the coarsening process. The issue is a smaller concern for H1′-based coarsening using BSIS since these methods produce small operator complexities that do not grow as the size of the problem is increased. In all cases, however, the number of processor boundary vertices relative to the number of interior vertices increases as the number of unknowns per processor decreases (e.g., on coarse levels). A few techniques may be applicable in this situation. The easiest solution is to simply use CLJP-c, or some other coarsening algorithm, on the coarser levels where few vertices are found. Although BSIS does not decrease in cost at this point, the total cost on the levels when BSIS is not used is low due to low complexity of coarse grids. A second approach is the application of a dynamic load balancing algorithm to increase the number of vertices on a processor (and, thus, decrease communication costs). If the number of vertices per processor per level is maintained at a high enough level, BSIS is still valuable on coarse grids. A third option is to replicate the operator matrix on processors, which leads to processors doing more of the same work, but by avoiding communication. The second and third ideas are similar in nature and both involve using dynamic load balancing techniques [23, 20, 21, 51, 22, 16].

Chapter 6

Parallel Compatible Relaxation

In contrast to the coarsening algorithms discussed in Chapters 3 through 5, compatible relaxation (CR) [8, 44, 10] does not utilize a strength of connection measure. CR methods instead use relaxation to identify smooth error.

6.1 Intuition and Design

The type of compatible relaxation discussed in this chapter is *concurrent CR* [44]. To select a coarse grid, smooth error is identified using CR iterations. An iteration of concurrent CR relaxes $Ae = 0$ at only F-points, while the unknowns at C-points are set to zero since CR assumes coarse-grid correction completely annihilates error at these unknowns.

The CR iteration identifies and forms a candidate set composed of vertices where smooth error is insufficiently damped. Independent sets are selected from candidate sets and added to the C-point set. The quality of relaxation is quantified by the *CR rate*, ρ_{cr}. A small CR rate indicates that relaxation on the F-points is effective, and that the coarsening process on that level should be terminated. Algorithm 6.1 shows the CR algorithm from [10].

Theory states that there exists an "ideal" prolongation operator for a coarse grid with a fast CR rate. The ideal prolongator, used with the corresponding coarse grid, produces a multigrid solver with a small convergence factor [26, 27]. A major challenge in exploiting this theoretical result is to find practical approximations to the ideal prolongator since the ideal prolongator is typically impractical to construct and use. Moreover, it is not known under what conditions such approximations produce fast AMG solve phases. Approximations to the ideal interpolation operator are developed in [10] and use a trace-minimization method [56, 62] to determine the nonzero entries in P given a structural sparse approximation of the ideal prolongator.

The contribution in this thesis is a parallel implementation of the CR algorithm targeting the independent set algorithms used to select C-points from the candidate set (Line 10 of Algorithm 6.1).

Algorithm 6.1 Compatible Relaxation

1: $F \leftarrow \Omega$
2: $C \leftarrow \emptyset$
3: $e^{(0)} \leftarrow 1 + \text{rand}(0,\, 0.25)$
4: $\alpha = 0.7$
5: **repeat**
6: Perform ν CR iterations on F, where $e_f^{(0)} \leftarrow \mathbf{0} + (e^{(0)})_f$
7: $\rho_{cr} \leftarrow \frac{\|e_f^{(\nu)}\|_{A_{ff}}}{\|e_f^{(\nu-1)}\|_{A_{ff}}}$
8: **if** $\rho_{cr} \geq \alpha$ **then**
9: Form candidate set

$$U \leftarrow \left\{ i : \frac{|(e_f^{(\nu)})_i|}{\|e_f^{(\nu)}\|_\infty} \geq 1 - \rho_{cr} \right\}$$

10: $D \leftarrow$ Independent set of U
11: $C \leftarrow C \cup D$
12: $F \leftarrow F \setminus D$
13: **end if**
14: **until** $\rho_{cr} < \alpha$

Any algorithm in Figure 3.1, for instance, may be used to select the independent set. In the next section, experimental results are presented for two parallel CR implementations: one with CLJP as the independent set algorithm and one with PMIS as the independent set algorithm.

6.2 Experimental Results

The suite of experiments in Chapter 4 are now run using CR with CLJP (CR-CLJP) and CR with PMIS (CR-PMIS). The relaxation method in both CR algorithms is a hybrid Jacobi/Gauss-Seidel approach, where Gauss-Seidel is used in processor interiors and Jacobi is used across processor boundaries. For brevity, only two experiments are presented. The results for CLJP and PMIS are reproduced for comparison. See Appendix C and [3] for results and data for all experiments. In each experiment, ν (Line 6 of Algorithm 6.1) is five.

3D 7-point Laplacian

Recall the 3D Laplacian test problem:

$$\begin{aligned} -\Delta u &= 0 \quad \text{on } \Omega \qquad (\Omega = (0,1)^3), \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{6.1}$$

As in Section 4.5.3, the problem is discretized with finite differences to yield the common 7-point stencil. The problem is scaled to provide a $50 \times 50 \times 50$ grid (125,000 unknowns) to each processor for all trials. On 256 processors, the problem is on a $400 \times 400 \times 200$ grid, resulting in 32 million unknowns in the largest trial. The results for normalized setup time, convergence factor, operator complexity, and work per digit-of-accuracy are given in Figure 6.1.

It is not surprising that the CR methods exhibit similar performance compared to the corresponding independent set-based coarsening algorithms. The convergence factors, operator complexities, and work per digit-of-accuracy are similar for each pair of methods.

The difference is in the normalized setup times. The setup times for both CR methods grow more slowly as the number of processors is increased than for the corresponding independent set-based methods. In CR, the coarsening process typically terminates before CLJP and PMIS terminate since CR determines when relaxation is sufficiently fast in order to require another level of coarse-grid correction. In some cases, the coarsest grid selected by CR is large compared to the other methods saving time because the coarsest levels require the greatest amount of communication relative to computation.

Tower plots for the four methods are shown in Figure 6.2. The towers are visually quite similar, but the number of levels selected by the CR methods is slightly smaller.

3D Unstructured Laplacian

In this section, results are reported for the Laplacian problem (6.1) discretized on an unstructured mesh by the finite element method. The problem on a single processor contains approximately 211,000 unknowns. The largest problem is on 512 processors and has approximately 100 million unknowns, which gives an average of 198,000 unknowns per processor. The partition size data for this problem is shown in Figure 4.16. Normalized setup times, convergence factors, operator complexities, and work per digit-of-accuracy are reported in Figure 6.3.

As before, correlations exist between CR-CLJP and CLJP and between CR-PMIS and PMIS. CR methods select good coarse grids for problems where the strength matrix provided to independent set-based algorithms is inaccurate. However, for the Laplacian problems tested in this section the strength of connection measure is sufficient.

Figure 6.4 displays the tower plots for this experiment. The towers are similar for the correlated methods, but once again, CR selects fewer levels in both cases.

The purpose of the experiments in this section is to showcase properties of early parallel CR

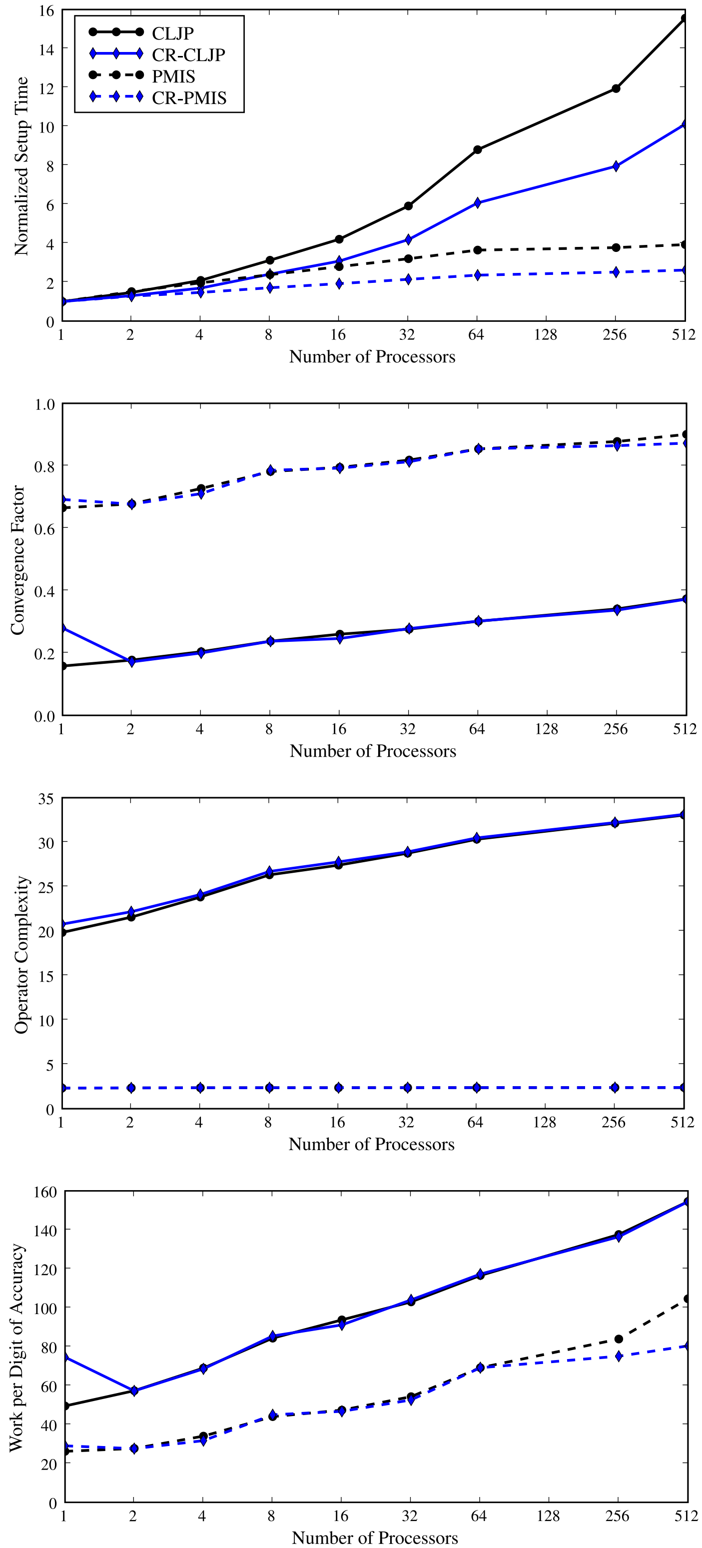


Figure 6.1: Results for the scaled 7-point Laplacian problem. The legend from the first plot applies to all four plots.

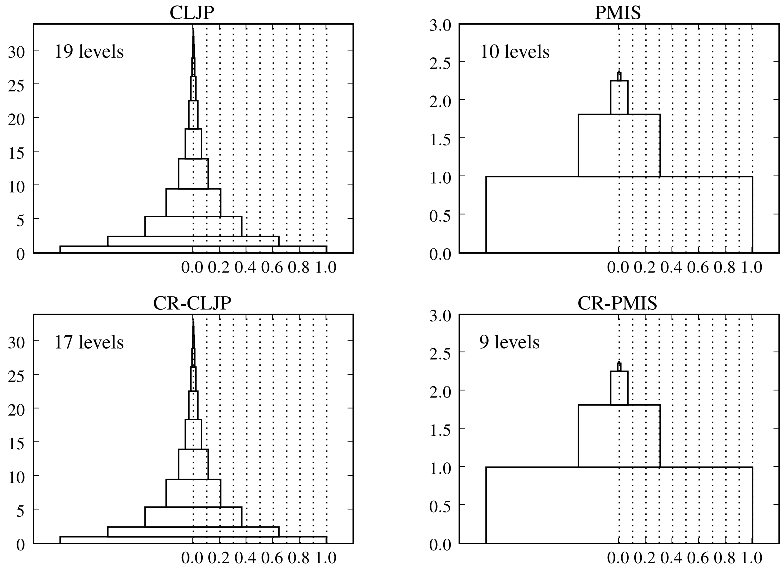


Figure 6.2: Tower plots for the 7-point Laplacian scaled problem. The towers shown are for the 512 processor trials. Notice the scale is not the same in each plot.

methods. To realize the full potential of CR methods, new prolongation operators are needed. As prolongation techniques advance, CR methods become applicable to a larger set of problems, including problems where strength of connection-based methods currently do not produce high-quality coarse grids.

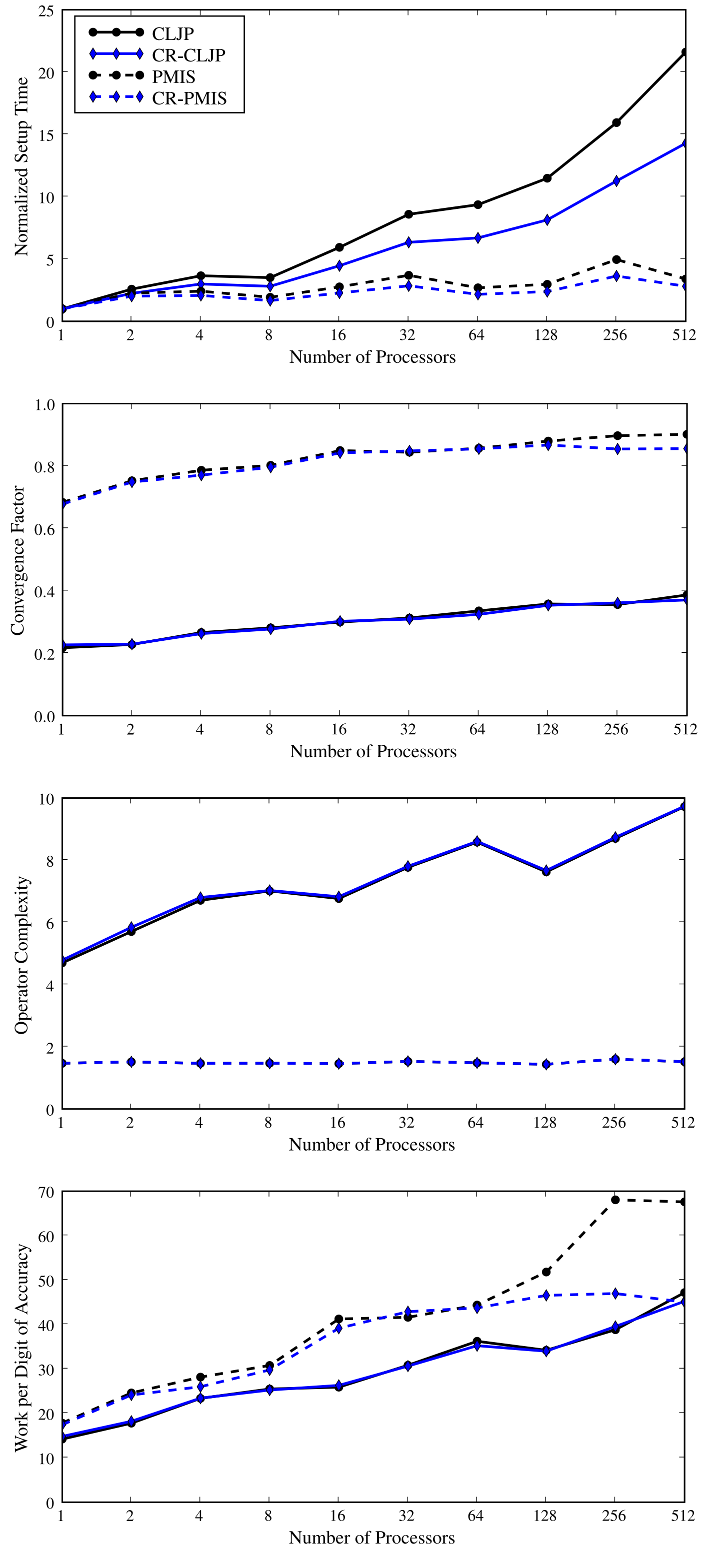


Figure 6.3: Results for the 3D unstructured Laplacian scaled problem. The legend from the first plot applies to all four plots.

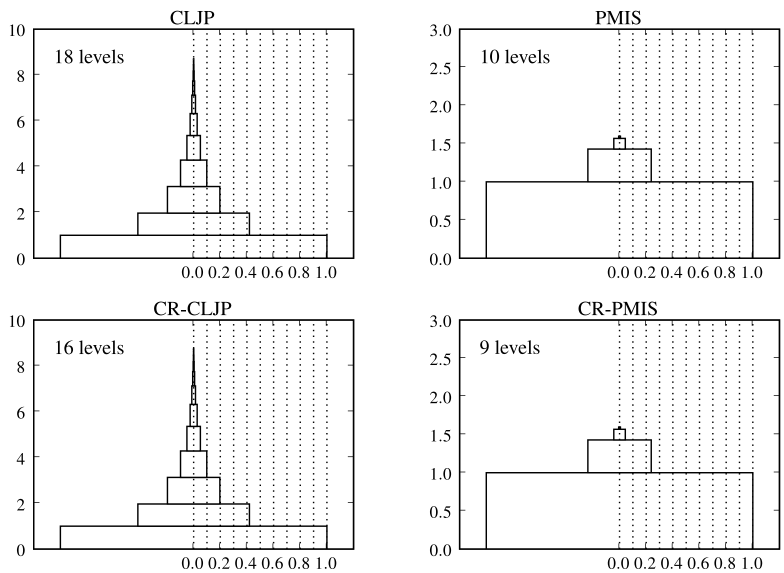


Figure 6.4: Tower plots for the 3D unstructured Laplacian scaled problem. The towers shown are for the 256 processor trials.

Chapter 7

Conclusions

This thesis focuses on the problem of coarse-grid selection for parallel algebraic multigrid and makes positive contributions by examining parallel coarse-grid selection from several perspectives. Following a review of the major contributions in the thesis, several future directions for parallel AMG are discussed.

7.1 Contributions

Analysis of CLJP. Differences in random augmentations given to vertices during CLJP initialization lead to large differences in the coarse grids produced. Analysis of random augmentations in Chapter 4 provides insight into the effects of applying random numbers to create parallelism. The results presented for modest sized problems demonstrate CLJP selects coarse grids much larger than those selected by RS. CLJP is, however, capable of producing identical coarse grids, which is the basis of CLJP-c.

Modification of CLJP to select with structure. The poor performance of CLJP on some types of problems results from its application of random augmentations to vertex weights. This thesis presents an algorithm called CLJP in Color (CLJP-c) in Chapter 4 that utilizes a different initial vertex weight to influence the method to select better coarse grids. The algorithm produces a graph coloring that is used while assigning initial vertex weights. Although coloring the graph requires more work than producing random weight augmentations, the savings outweigh the costs. For targeted problems, CLJP-c demonstrates large improvements in performance over CLJP.

Extension to H1′ algorithms: PMIS-c1 and PMIS-c2. The PMIS coarsening algorithm is the H1′ counterpart to CLJP. The ideas used to produce CLJP-c are also applied to PMIS in Chapter 4. The direct application of the technique leads to PMIS-c1.

The PMIS algorithm selects a converse dominating set and, given an appropriate set of random weight augmentations, selects a converse dominating set with the smallest number of vertices possible. In such a coarse grid, most C-points are three hops from their nearest C-point neighbors. PMIS-c2 is designed to produce a coarse grid that exploits this distance by using a distance-two coloring routine rather than the usual graph coloring algorithm.

Parallel compatible relaxation methods. Compatible relaxation (CR) methods select coarse grids that are "compatible" with the relaxation method. The relaxation method is used to identify smooth error and then select a coarse grid to accurately represent that error. The CR approach is advantageous since smooth error is defined based on the relaxation method. The contribution in this thesis to CR research is a parallel algorithm in Chapter 6.

Extensive experimental results. Several experiments are executed using the coarsening algorithms discussed in this thesis. This experimental collection constitutes the single largest set of published experiments for the coarsening problem. A set of results with substantial amounts of data is presented in Chapter 4, with additional experiments in Chapters 5 and 6. It is impractical to present all experimental data, due to its size, so Appendix C contains more data on the experiments in tabular form.

Novel analysis tools. In addition to using global measures to study the performance of the setup phase, new tools are used to aid in analysis and understanding of coarsening algorithms. Viewing the progress of a coarsening algorithm on a per level basis is useful for gaining insight into how the algorithm coarsens during different parts of the process. The tower plots in Chapter 4 offer a view of the grid and operator complexities produced on each level. Throughout this research, visualization tools provide insight into the coarsening properties of the algorithms studied.

Improved search for coarse-grid selection. Independent set-based algorithms select coarse grids by first setting up data structures, coloring the graph (if applicable), and initializing weights. Next, independent sets are selected, followed by an update to vertex weights. Independent set selection determines new C-points, whereas vertex weight updates determine new F-points. Independent sets are selected by searching the graph for vertices satisfying (5.1). The research that produces CLJP-c and related algorithms enables the development of a more efficient search step for coarsening. By keeping vertices sorted in buckets, search is executed without the need for weight comparisons between adjacent vertices. The BSIS algorithm presented in

Chapter 5 employs this bucket method to select coarse grids based on heuristic H1.

Invariance in coarse-grid selection. Theoretical results are presented in Chapter 5 proving all algorithms using generalized conditions select identical coarse grids, given the same weights. The result is both powerful and useful because it creates opportunities to develop more efficient coarse-grid selection algorithms without changing the results of the coarsening.

BSIS update aggregation. The bucket data structure used by BSIS must be updated as vertex weights change. Standard BSIS updates the data structure following any weight update, summing to significant cost over the total coarsening process. It is observed in this thesis that accuracy of the data structure is not altogether necessary. Rather, BSIS only needs to be capable of determining which vertices are in the proper location in the data structure. Developed in Chapter 5, aggregate weight update takes a lazy approach to updating the data structure and puts off doing work until required. Vertices are moved to the correct location only when they are in the non-empty bucket with largest weight (i.e., the bucket containing vertices to be added to C), yielding large performance gains over the original BSIS algorithm, which is already a large improvement over CLJP-c.

The experimental results presented show more than 15% gains in BSIS over CLJP-c, and BSIS with aggregate weight update exhibits nearly 25% reduction in cost over CLJP-c.

7.2 Future Work

The algorithms developed and theory established in this thesis form a base for further research. Several possible future directions are listed below.

Algorithmic improvements to vertex update. Updating vertex weights is a significant source of computational cost in coarse-grid selection, and improvements made to update routines are certain to contribute to the efficiency of coarse-grid selection algorithms. The search routines in coarse-grid selection are studied in Chapter 5, and the improvements made and insight gained are applicable to this problem.

Apply lessons learned to prolongator construction algorithms. Prolongation and coarse-grid selection are closely coupled processes. Traditional prolongators depend on H1 or H1′, so many of the techniques in use share similarities. Advances made in developing better vertex weight update algorithms are applicable to prolongator construction.

Parallel BSIS (*in preparation*). The performance gains observed when using BSIS motivate further investigation into parallel coarse-grid selection with BSIS. Preliminary results are encouraging.

Combined coarsening-prolongation algorithm. Coarse-grid selection and prolongator construction use the same graph as input. Furthermore, both rely on many of the same graph traversals. These processes, however, are decoupled in implementation. The advantages of using a single algorithm to coarsen and construct the prolongator simultaneously include avoiding the traversal of the graph multiple times for the same information and having a coarsening routine that is aware of the impact to operator complexity as it selects C-points.

Continue bringing graph algorithm perspective to the problem. The study of coarse-grid selection as a combinatorial problem provides insight that is otherwise potentially overlooked. Combinatorial scientific computing is a thriving field, and research from other areas may provide solutions for AMG. Graph coloring is particularly closely related to coarse-grid selection, but other areas, such as matrix ordering for sparse direct methods [33, 34, 46], also provide insight.

Design for new architectures. Parallel architectures are currently undergoing a transformation. Within the last few years, physical and power constraints have forced the architecture industry to increase desktop computing power through the use of new designs. Multi-core technology and the Cell architecture have both appeared in the commercial market in the last seven years [35]. Although the traditional approach in scientific computing is to run a single process per processor, new parallel algorithms target thread-level parallelism as multi-core chips become more pervasive. The study of algorithm design for multi-core architectures is a timely pursuit for AMG development.

Machine learning. The possibility of applying machine learning to a number of problems in AMG exists. Strength threshold θ, for example, is often 0.25 unless a cause for change is perceived. Finding the optimal setting for θ is largely unstudied and differs for many input matrices. Similar opportunities exist for smoother parameters or in selecting a coarsening algorithm. The sheer number of possibilities is intimidating, but for many AMG parameters, machine learning is a potentially beneficial approach.

Dynamic load balancing. As a multigrid cycle progresses to coarser levels, the number of unknowns per processor decreases. Consequently, the number of processor boundary unknowns increases relative to the number of interior unknowns, leading to increased communication relative to computation. Coarse-grid selection is affected since each iteration of CLJP, CLJP-c, PMIS, etc., is followed by a communication step. The study of dynamic load balancing is active and offers the potential of significantly cutting communication costs. Through data migration or data replication, the time needed on coarser levels in AMG (for both the setup phase and solve phase) is expected to decrease.

7.3 Closing Remarks

The study of numerical linear solvers is an interesting and important endeavor, and parallel solvers continue to be an important area of research. The solution of linear systems, such as those arising from the discretization of partial differential equations, is an integral aspect of scientific computing. AMG emerged around twenty years ago [48] and has since undergone rapid progress. The research of coarse-grid selection algorithms for parallel AMG is relatively new, with CLJP being published fewer than ten years before this thesis [18]. Since that time, many new coarsening algorithms have been developed, and the contributions presented in this thesis add to the progress, bringing new perspectives and ideas to the study of efficient parallel AMG.

Appendix A

Geometric Multigrid and the Two-Grid Operator

Geometric multigrid and the two-grid operator are introduced in this appendix. Stencil notation is introduced followed by an overview of the concepts in multigrid. Finally, there is a discussion on solving systems of PDEs with multigrid. See [14, 55] for a complete introduction.

Geometric multigrid functions on geometries based on regular grids, such as Cartesian grids. Nested coarse grids implicitly exist for such geometries and are used to construct coarse-grid problems.

A.1 Stencil Notation

Many linear solvers operate globally on a matrix, but this is not the case in multigrid. Instead, multigrid works locally on each grid point, which allows operators (i.e., matrices) in multigrid to be defined using compact *stencil operators*.

A stencil defines an operator for the local interaction of each grid point, where stencils considered in this thesis are of *five-point* or *nine-point* form. For example, a common five-point stencil is that of the 2D Poisson equation,

$$-\Delta u = f, \tag{A.1}$$

discretized using finite differences yielding the linear system $Ax = b$. The matrix A is alternatively represented by the stencil produced through the discretization of (A.1) is

$$-\Delta = \frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}, \tag{A.2}$$

where h is the distance between neighboring grid points. This stencil defines that an unknown $x_{i,j}$

corresponding to an interior point in row i, column j of the grid is defined as

$$-\Delta x_{i,j} = \frac{1}{h^2}\left(4x_{i,j} - x_{i-1,j} - x_{i+1,j} - x_{i,j-1} - x_{i,j+1}\right).$$

A.2 Basic Concepts

Multigrid methods rely on two processes: *relaxation* (also called *smoothing*) and *coarse-grid correction*. The properties of these processes allow each to be successfully applied toward solving a linear system by complementing the weaknesses of the other.

Relaxation quickly annihilates *high-frequency error* and leaves *low-frequency error* relatively unchanged. The relaxation procedure in geometric multigrid and in algebraic multigrid is the same, and the basic ideas of relaxation are developed in Section 2.1.1.

In geometric multigrid, low-frequency error corresponds to smooth error modes. That is, low-frequency error is smooth and varies slowly locally (see Figure 2.3(a)). Smooth error is approximated accurately with fewer degrees of freedom on the coarse grid, and a coarse-grid correction process is applied to remove the low-frequency error. The concept of coarse-grid correction is discussed in Section 2.1.2.

A.3 Components of Multigrid

A multigrid method is constructed from five fundamental components: the *smoother*, the *coarsening strategy*, the *coarse-grid operator*, the *restriction operator*, and the *prolongation operator*. Each component is discussed below, and common examples are given.

A.3.1 Smoothers

Smoothers are used to quickly dampen high-frequency errors and are also often called relaxation methods. Any iterative method exhibiting this high-frequency damping property is a smoother.

There are several classes of smoothers, such as *point smoothers*, *line smoothers*, and *block smoothers*. Point smoothers are the most basic and include methods like *weighted-Jacobi iteration* and *Gauss-Seidel iteration*.

A.3.2 Coarsening Strategy

The coarse grids geometric multigrid uses largely depend on the problem to be solved. *Standard coarsening* doubles the distance between degrees of freedom. That is, if h is the distance between unknowns on the fine grid, then $2h$ is the distance between unknowns on the coarse grid. Figure 2.1 illustrates standard coarsening.

Standard coarsening is not always the most effective strategy. *Anisotropic* problems have much stronger coupling between the grid points in one direction than in another. For example,

$$-u_{xx} - \epsilon u_{yy} = f \tag{A.3}$$

is anisotropic when $0 < \epsilon \ll 1$. Standard coarsening is ineffective in this case, and multigrid converges more quickly if *semi-coarsening* is used. By coarsening only in directions that are strongly connected, coarse-grid correction accurately captures the smooth error of (A.3).

Other coarsening strategies are important in practice, but the remainder of this introduction assumes standard coarsening is used.

A.3.3 Restriction

Restriction operators transfer residuals to the next coarse grid. In stencil notation the restriction operator is denoted by R.

Common restriction operators are *injection*, *full weighting*, and *half weighting*. Injection is the simplest restriction technique. In injection, a coarse-grid point has the same value on both the fine and coarse grid. Although this restriction is simple, it typically does not represent the residual well on the coarse grid.

Full weighting restriction is more commonly used and in general, works more effectively. For full weighting,

$$R = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_h^{2h}, \tag{A.4}$$

which computes the value at a coarse-grid point with a weighted average of fine points around it. In full weighting, the value of coarse-grid points are more indicative of the residual on the fine grid.

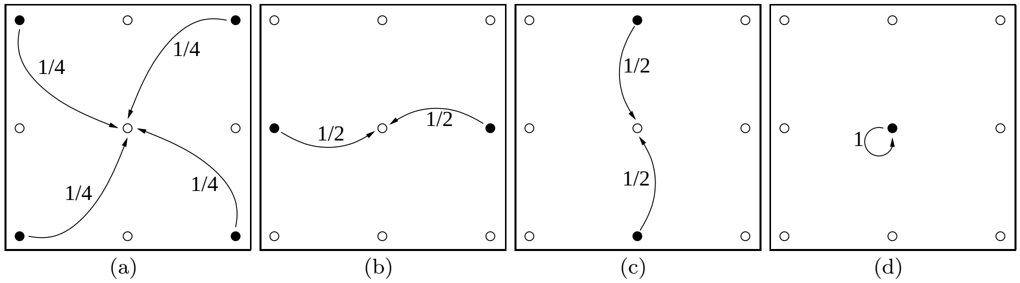


Figure A.1: Bilinear interpolation for each of four points. A point on the fine grid relates to the coarse grid in one of four ways. Each figure illustrates one of these situations and the weighting for the prolongation. Coarse-grid points are denoted as filled circles, and fine-grid points are empty circles. Notice the relationship between these weights and the prolongation operator (A.5).

A.3.4 Prolongation

Prolongation is the opposite process to restriction. It transfers a vector onto the next fine grid. Prolongation can in some cases be called *interpolation*. The prolongation operator is denoted by P.

For two-dimensional grids, a common prolongation operator is the *bilinear interpolation* operator,

$$P = \frac{1}{4} \left] \begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix} \right[_{2h}^{h}, \tag{A.5}$$

which is a scaled transpose of full weighting. Fine-grid points are assigned averages of the adjacent coarse-grid points. Bilinear interpolation is illustrated in Figure A.1.

A.3.5 Coarse-Grid Operator

To solve the defect equation with coarse-grid correction a coarse-grid operator must first be selected. One possibility is to re-discretize the original problem on the coarse grid. A common alternative is the *Galerkin operator*,

$$A_H = RA_hP, \tag{A.6}$$

where A_h and A_H are the fine-grid matrix and coarse-grid matrix, respectively. The Galerkin operator has several advantages making it attractive for use as the coarse-grid operator. One advantage is a natural averaging of the coefficients, making it particularly appropriate for problems with discontinuous coefficients or variable coefficients [57].

The Galerkin operator also has some disadvantages. In some situations, implementing the Galerkin operator proves more difficult and inappropriate than rediscretizing the system. Also, the Galerkin product tends to "enlarge" the stencil. That is, a five-point stencil on the fine grid in two dimensions tends to become a nine-point stencil on coarse grids when the Galerkin coarse-grid operator is used [55].

In practice, both the Galerkin operator and the operator resulting from rediscretizing the system are commonly used.

A.4 Assembling the Two-Grid Operator

Individual components of multigrid are described in Section A.3. In this section those components are assembled into a simple two-grid cycle, and an expression representing the two-grid cycle is derived. One sweep of a two-grid cycle consists of three steps: presmoothing, coarse-grid correction, and postsmoothing.

A.4.1 Presmoothing

The first step of a two-grid cycle is to smooth fine-grid points with ν_1 passes of the relaxation method. As discussed in Section A.2, a number of different smoothers are available to use. Presmoothing is expressed as

$$\hat{x}_1^k = S_h^{\nu_1} x^k + \tilde{c}_1, \tag{A.7}$$

where S_h is the iteration matrix and $\tilde{c}_1$ is a vector depending only on the right-hand side, the splitting of A, and ν_1 (see Section 2.1.1).

A.4.2 Coarse-Grid Correction

The second step of the two-grid method is coarse-grid correction. This starts by calculating the residual,

$$r_h = b - A_h \hat{x}_1^k, \tag{A.8}$$

which is then transferred to the coarse grid using the restriction operator, producing the coarse-level residual:

$$r_H = R r_h. \tag{A.9}$$

Following restriction, the defect equation on the coarse grid,

$$A_H e_H = r_H, \tag{A.10}$$

is solved by a direct method. The coarse-level error e_H is interpolated to the fine grid to yield the fine-level error:

$$e_h = Pe_H. \tag{A.11}$$

Finally, the approximate solution on the fine grid is updated by computing

$$\hat{x}_2^k = \hat{x}_1^k + e_h. \tag{A.12}$$

By combining the results from (A.8) through (A.12), a single operator representing coarse-grid correction is derived. This coarse-grid operator is assembled starting with (A.12) and substituting the right-hand sides of (A.8) through (A.11),

$$\begin{aligned} \hat{x}_2^k &= \hat{x}_1^k + PA_H^{-1}R(b - A_h\hat{x}_1^k) \\ &= (I_h - PA_H^{-1}RA_h)\hat{x}_1^k + PA_H^{-1}Rb, \end{aligned} \tag{A.13}$$

where I_h is the identity operator. Letting $K_h^H = (I_h - PA_H^{-1}RA_h)$ and $\hat{c} = PA_H^{-1}Rb$ this becomes

$$\hat{x}_2^k = K_h^H\hat{x}_1^k + \hat{c}, \tag{A.14}$$

where K_h^H is the *coarse-grid operator*.

A.4.3 Postsmoothing

The two-grid cycle finishes with ν_2 sweeps of the smoother on the fine grid. Postsmoothing is mathematically identical to presmoothing. Note that the same smoothing operator (S_h) has been used here because typically presmoothing and postsmoothing are done with the same smoother. Therefore, postsmoothing is expressed as

$$x^{k+1} = S_h^{\nu_2}\hat{x}_2^k + \tilde{c}_2. \tag{A.15}$$

A.4.4 Two-Grid Operator

The expressions for presmoothing, coarse-grid correction, and postsmoothing are now combined to form the two-grid operator, starting with the expression for postsmoothing in (A.15). Replacing $\hat{x}_2^k$

from (A.15) with the right side of (A.14) yields

$$x^{k+1} = S_h^{\nu_2}(K_h^H \hat{x}_1^k + \hat{c}) + \tilde{c}_2. \tag{A.16}$$

The presmoothing expression (A.7) defines $\hat{x}_1^k$. Substitution gives

$$x^{k+1} = S_h^{\nu_2}[K_h^H(S_h^{\nu_1} x^k + \tilde{c}_1) + \hat{c}] + \tilde{c}_2. \tag{A.17}$$

Let $\bar{c} = S_h^{\nu_2} K_h^H \tilde{c}_1 + S_h^{\nu_2}\hat{c} + \tilde{c}_2$ and $M_h^H = S_h^{\nu_2} K_h^H S_h^{\nu_1}$, reducing the two-grid cycle to the form of a stationary iterative method,

$$x^{k+1} = M_h^H x^k + \bar{c}, \tag{A.18}$$

where $\bar{c}$ depends only on the operators and the right-hand side (i.e., $\bar{c}$ is a constant term). The operator M_h^H is the *two-grid operator*.

Significance of the two-grid operator

The two-grid operator is the foundation of a local Fourier analysis technique called two-grid analysis (see Appendix B.3). Also, useful multigrid theory depends on the two-grid cycle. For instance, if a two-grid method converges independent of h, then theory shows a W-cycle exhibits similar convergence properties.

Convergence of the two-grid method depends only on M_h^H. This is shown by starting with (A.18) and subtracting the true solution from both sides:

$$u^* - u^{k+1} = M_h^H u^* + \bar{c} - (M_h^H u^k + \bar{c}). \tag{A.19}$$

Further manipulation yields $e^{k+1} = M_h^H e^k$. This is expressed for the error in any iteration as

$$e^k = (M_h^H)^k e^0. \tag{A.20}$$

The magnitude of the error decreases in each iteration if and only if the spectral radius of M_h^H is less than one.

Appendix B

Local Fourier Mode Analysis

Local mode analysis (LMA) is used to estimate the performance of smoothers and multigrid. LMA replaces errors in an iteration with Fourier modes and derives a growth factor based on how effectively the error modes are damped in each iteration. This growth factor is useful when the slowest converging mode is known because it gives an upper bound on the decay rate for all modes.

A basic example demonstrating LMA is provided in Section B.1. In Section B.2, the procedure for calculating a smoothing factor is shown. Two-grid analysis is introduced in Section B.3, and LMA with different smoothers is discussed in Section B.4.

The concepts presented in this appendix are at a level appropriate to introduce LMA. For a more complete introduction to LMA, see [58, 60, 61, 55].

B.1 Basic Idea

In this section, the effectiveness of solving linear systems $Ax = b$ with stationary iterative methods is analyzed. Stationary iterative methods split A into two matrices M and N such that

$$A = M - N. \tag{B.1}$$

Further manipulation of the linear system yields

$$Mx = Nx + b. \tag{B.2}$$

The solution of the linear system is a *stationary point* of the iterative method

$$Mx^{m+1} = Nx^m + b, \tag{B.3}$$

where the superscripts on x denote iteration number.

Take for example the 2D Poisson equation, $-\Delta u = f$ discretized by finite differences to yield the linear equation

$$\frac{-x_{j+1,k} - x_{j-1,k} + 4x_{j,k} - x_{j,k+1} - x_{j,k-1}}{h^2} = b_{j,k} \tag{B.4}$$

for the unknown of the point in row j, column k of the grid. Furthermore, assume that this discretization is to be solved using lexicographic Gauss-Seidel (GS-LEX). For GS-LEX, M is the lower triangle with diagonal, and $-N$ is the upper triangle without diagonal. For this analysis, only the application of the solver on a single interior point of the grid is considered. The iteration for an interior point becomes

$$x_{j,k}^{m+1} = \frac{1}{4}\left(x_{j+1,k}^{m} + x_{j-1,k}^{m+1} + x_{j,k+1}^{m} + x_{j,k-1}^{m+1}\right) + \frac{h^2}{4}\left(b_{j,k}\right). \tag{B.5}$$

This iteration is manipulated to yield an error iteration. Define the error in iteration $m+1$ as $e^{m+1} = x^* - x^{m+1}$, where x^* is the true solution. Subtracting the true solution iteration from (B.5) yields

$$\begin{aligned} x^* - x_{j,k}^{m+1} &= \frac{1}{4}\left(x_{j+1,k}^{*} + x_{j-1,k}^{*} + x_{j,k+1}^{*} + x_{j,k-1}^{*}\right) - \frac{h^2}{4}\left(b_{j,k}\right) \\ &\quad - \frac{1}{4}\left(x_{j+1,k}^{m} + x_{j-1,k}^{m+1} + x_{j,k+1}^{m} + x_{j,k-1}^{m+1}\right) + \frac{h^2}{4}\left(b_{j,k}\right). \end{aligned}$$

This expression simplifies to

$$e_{j,k}^{m+1} = \frac{1}{4}\left(e_{j+1,k}^{m} + e_{j-1,k}^{m+1} + e_{j,k+1}^{m} + e_{j,k-1}^{m+1}\right). \tag{B.6}$$

Equation (B.6) is an iterative expression of the error in the approximate solutions. Once again, the premise of Fourier analysis is to replace the errors in an error iteration by Fourier modes, where the form of a Fourier mode is

$$e_{j,k}^{m} = T(m)e^{i(j\theta_1 + k\theta_2)}, \tag{B.7}$$

where $T(m)$ is the amplitude of the mode in the mth iteration. Substituting Fourier modes into (B.6) gives

$$\begin{aligned} T(m+1)e^{i(j\theta_1+k\theta_2)} &= \frac{1}{4}\Big(T(m)e^{i((j+1)\theta_1+k\theta_2)} + T(m+1)e^{i((j-1)\theta_1+k\theta_2)} + \\ &\quad + T(m)e^{i(j\theta_1+(k+1)\theta_2)} + T(m+1)e^{i(j\theta_1+(k-1)\theta_2)}\Big). \end{aligned} \tag{B.8}$$

To see the damping properties of the method, the change in the amplitude from iteration to iteration is examined. This is done by transforming (B.8) into the form

$$T(m+1) = \tilde{S}_h(\boldsymbol{\theta})T(m), \tag{B.9}$$

where $\boldsymbol{\theta} = (\theta_1, \theta_2)$. In this form, (B.8) becomes

$$T(m+1) = \left(\frac{e^{i\theta_1} + e^{i\theta_2}}{4 - e^{-i\theta_1} - e^{-i\theta_2}}\right) T(m), \tag{B.10}$$

where

$$\tilde{S}_h(\boldsymbol{\theta}) = \frac{e^{i\theta_1} + e^{i\theta_2}}{4 - e^{-i\theta_1} - e^{-i\theta_2}}. \tag{B.11}$$

$|\tilde{S}_h(\boldsymbol{\theta})|$ is called an *amplification factor* for the mode $\boldsymbol{\theta}$, and $\tilde{S}_h$ is the *symbol* for the amplification factor. By looking at the value of $|\tilde{S}_h(\boldsymbol{\theta})|$ for different modes (different $\boldsymbol{\theta}$), the slowest modes to be damped by the iterative method are discovered. The rate at which the slowest mode is damped approximates the asymptotic convergence factor for solving the problem, where the initial guess contains that error mode. Due to periodicity, only the modes from $[-\pi, \pi) \times [-\pi, \pi)$ need to be examined to discover the mode with the slowest decay.

Figure B.1 shows the amplification factors for this example. Observe the maximum value of $|\tilde{S}_h(\boldsymbol{\theta})|$ is nearly one, meaning the corresponding mode is barely damped from one iteration to the next, so convergence is extremely slow.

B.2 Smoothing Analysis

Smoothing analysis is used for estimating the convergence factor of multigrid. An important concept in LMA is the distinction between *low-frequency modes* and *high-frequency modes*. These definitions are made on the domain $[-\pi, \pi) \times [-\pi, \pi)$. Low-frequency modes are defined as

$$\boldsymbol{\theta} \in T^{\text{low}} := \left[-\frac{\pi}{2}, \frac{\pi}{2}\right) \times \left[-\frac{\pi}{2}, \frac{\pi}{2}\right). \tag{B.12}$$

The high-frequency modes lie in the remainder of the domain. That is,

$$\boldsymbol{\theta} \in T^{\text{high}} := [-\pi, \pi) \Big\backslash \left[-\frac{\pi}{2}, \frac{\pi}{2}\right) \times [-\pi, \pi) \Big\backslash \left[-\frac{\pi}{2}, \frac{\pi}{2}\right). \tag{B.13}$$

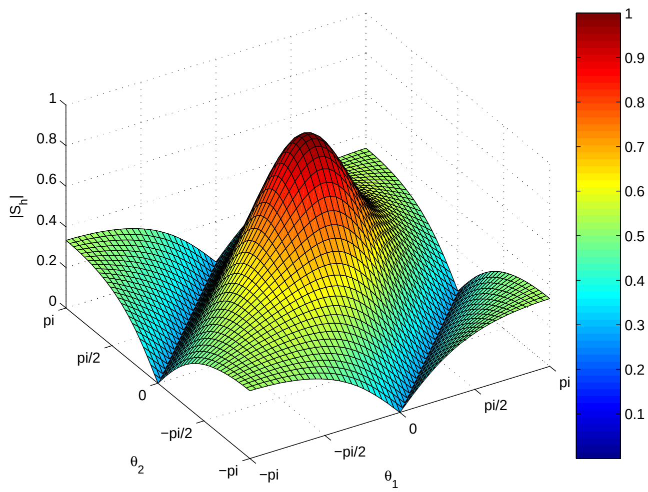


Figure B.1: Fourier analysis of GS-LEX applied to Poisson's equation.

Figure B.2 illustrates the high-frequency and low-frequency domains. Note that these definitions are for full coarsening. In different coarsening strategies the definitions of high- and low-frequency modes changes.

Smoothing analysis works by considering the damping effects of a smoother on high-frequency modes while assuming all low-frequency error are annihilated by coarse-grid correction. It also assumes coarse-grid correction has no effect on high-frequency modes.

The result from smoothing analysis is the *smoothing factor*. The smoothing factor, μ, for the smoothing operator S_h is defined as

$$\mu(S_h) = \sup\{|\tilde{S}_h(\boldsymbol{\theta})| : \boldsymbol{\theta} \in T^{\text{high}}\}. \tag{B.14}$$

Smoothing analysis for Poisson's equation relaxed by GS-LEX is identical to the example from Section B.1 except the values of $\tilde{S}_h(\boldsymbol{\theta})$ for $\boldsymbol{\theta} \in T^{\text{low}}$ are discarded. This gives a smoothing factor of 0.5 for this problem, which means the error is damped by one half in each multigrid cycle. The results of smoothing analysis for the model problem are shown in Figure B.3.

B.3 Two-Grid Analysis

The information provided by smoothing analysis is a very rough estimate of convergence factor. A better estimate is usually found by Fourier analysis on coarse-grid corrections, in addition to the

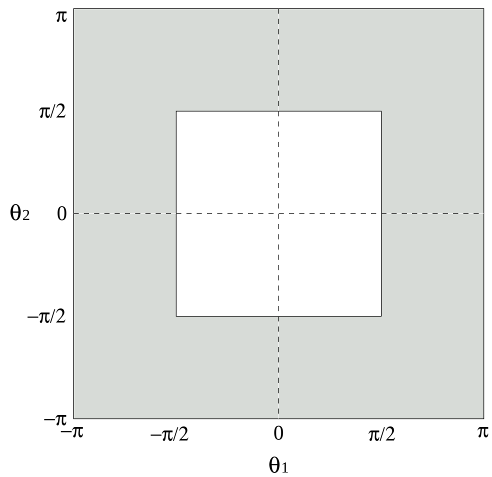


Figure B.2: High- and low-frequency regions on $[-\pi, \pi) \times [-\pi, \pi)$. The low-frequency modes lie in the white interior region of the domain, and the high-frequency modes lie in the shaded region.

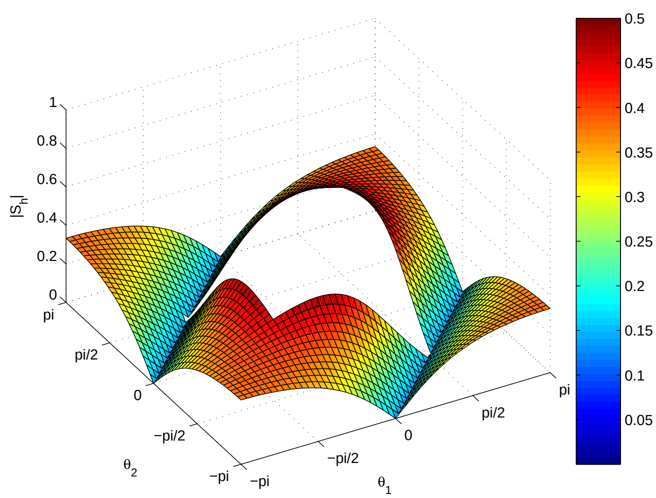


Figure B.3: Smoothing analysis of GS-LEX applied to Poisson's equation.

smoothing steps. This type of analysis is called *two-grid analysis.*

Smoothing analysis seeks the amplification factor of the smoothing operator, S_h. Similarly, in two-grid analysis, the amplification factor of the two-grid operator,

$$M_h^{2h} = S_h^{\nu_2} K_h^{2h} S_h^{\nu_1}, \tag{B.15}$$

is calculated. The derivation of the two-grid operator is shown in Appendix A.4. The number of presmoothing sweeps is ν_1, and ν_2 is the number of postsmoothing sweeps. K_h^{2h} is the coarse-grid operator,

$$K_h^{2h} = I_h - P A_H^{-1} R A_h, \tag{B.16}$$

where I_h is the identity matrix.

Two-grid analysis proceeds by looking at groups of four modes on the fine grid that appear identical on the coarse grid. That is, these four modes *alias* to the same mode on the coarse grid because fewer modes are representable on the coarse grid. As explained in [55], these modes are of the form

$$\begin{aligned} \boldsymbol{\theta}^{(0,0)} &:= (\theta_1, \theta_2), \qquad & \boldsymbol{\theta}^{(1,1)} &:= (\bar{\theta}_1, \bar{\theta}_2), \\ \boldsymbol{\theta}^{(1,0)} &:= (\bar{\theta}_1, \theta_2), \qquad & \boldsymbol{\theta}^{(0,1)} &:= (\theta_1, \bar{\theta}_2), \end{aligned} \tag{B.17}$$

$$\bar{\theta}_i := \begin{cases} \theta_i + \pi & \text{if } \theta_i < 0, \\ \theta_i - \pi & \text{if } \theta_i \geq 0. \end{cases}$$

By looking at four modes at a time the symbols become 4×4 as opposed to scalar, as in smoothing analysis. The symbols of this form are denoted by typographic "hats" above the symbol. For example, the symbol for the smoother used in two-grid is

$$\hat{S}_h = \begin{pmatrix} \tilde{S}_h(\boldsymbol{\theta}^{(0,0)}) & & & \\ & \tilde{S}_h(\boldsymbol{\theta}^{(1,1)}) & & \\ & & \tilde{S}_h(\boldsymbol{\theta}^{(1,0)}) & \\ & & & \tilde{S}_h(\boldsymbol{\theta}^{(0,1)}) \end{pmatrix}. \tag{B.18}$$

Two-grid analysis involves finding the symbols $\hat{S}_h^{\nu_2}(\boldsymbol{\theta})$, $\hat{K}_h^{2h}(\boldsymbol{\theta})$, and $\hat{S}_h^{\nu_1}(\boldsymbol{\theta})$. Once that information has been acquired, the maximum of $\hat{M}_h^{2h}(\boldsymbol{\theta})$ is found by testing various values of $\boldsymbol{\theta}$.

How to find the symbols of smoothing operators is shown in previous sections, leaving only

$\hat{K}_h^{2h}(\boldsymbol{\theta})$ to discuss. As stated in [55], $\hat{K}_h^{2h}(\boldsymbol{\theta})$ is a 4×4 matrix representing K_h^{2h}:

$$\hat{K}_h^{2h}(\boldsymbol{\theta}) = \hat{I}_h - \hat{P}(\boldsymbol{\theta})(\hat{A}_H(2\boldsymbol{\theta}))^{-1}\hat{R}(\boldsymbol{\theta})\hat{A}_h(\boldsymbol{\theta}). \tag{B.19}$$

For this presentation, it is not particularly important to understand exactly what each symbol looks like. It is important to realize that two-grid analysis is simply the result of taking the two-grid operator in Appendix A.4 and replacing each component with Fourier modes. This method gives sharp estimates for the convergence factor of multigrid on some problems. Cases exist, however, where two-grid analysis does not work particularly well. One way to obtain better results is to use more than one level of coarse-grid correction in the analysis.

B.4 Using Other Smoothers

LMA is compatible with many smoothers. Doing the analysis with any smoother that uses a splitting of the stencil in terms of a single point, such as GS-LEX or Jacobi smoothing, is straightforward.

Smoothers using a simple splitting are not the only popular options. Gauss-Seidel red-black (GS-RB) is another commonly used smoother. It is possible to do LMA with GS-RB, and similar smoothers like 4-COLOR smoothing, using a more complicated analysis. For a discussion on doing LMA with GS-RB, see [55].

B.5 Computational LMA

Computational packages for conducting Fourier analysis [1, 59] provide multigrid users with powerful tools. Computational LMA enables the determination of expected performance for many smoothers to be quickly computed, which provides opportunities to save computation time by determining an efficient multigrid configuration prior to solving the problem.

Appendix C

Additional Experimental Results

The purpose of this appendix is to provide additional experimental data from the experiments in Section 4.5. All data presented in the appendix appears in tabular form to provide interested parties with numerical results. The tables contain information on the trials, the relative problem size for each trial, absolute and relative grid and operator complexities, absolute and relative amounts of work per digit-of-accuracy, convergence factors, absolute and relative setup times, and information detailing the number of degrees of freedom and unknowns in the operator matrices on all levels for selected trials.

C.1 Fixed-Size 3D 7-Point Laplacian

This section reports the results for experiments on a strongly scaled 7-point Laplacian problem. The problem is defined as

$$\begin{aligned} -\Delta u &= 0 \quad \text{on } \Omega \qquad (\Omega = (0,1)^3), \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{C.1}$$

where the Laplacian is discretized using finite differences yielding the standard 7-point stencil. The individual trials and problem sizes for each trial are listed in Table C.1. The domain in all trials is a $128 \times 128 \times 128$ grid, which gives approximately two million unknowns.

The data is organized into the following tables.

Trial information .... Table C.1

Grid complexities .... Table C.2

Relative grid complexities .... Table C.3

Operator complexities .... Table C.4

Relative operator complexities .... Table C.5

Amount of work per digit-of-accuracy .... Table C.6

Relative amount of work per digit-of-accuracy .... Table C.7

Convergence factors .... Table C.8

Setup times .... Table C.9

Relative setup times .... Table C.10

Level-by-level degrees of freedom .... Tables C.11 and C.12

Level-by-level nonzeros .... Tables C.13 and C.14

p	1	2	4	8	16	32	64	128	256
Relative n	1	1	1	1	1	1	1	1	1

Table C.1: Trials and relative trial sizes for the strongly scaled 7-point Laplacian. The number of processors (p) is shown in the first row, and the number of unknowns (n) relative to the number of unknowns in the smallest trial is shown in the second row. This problem is strongly scaled, meaning the problem size is unchanged as the number of processors increases.

p	1	2	4	8	16	32	64	128	256
Falgout	1.64	1.64	1.64	1.64	1.65	1.65	1.65	1.66	1.68
CLJP	2.44	2.44	2.44	2.44	2.44	2.44	2.44	2.44	2.44
CLJP-c	1.66	1.67	1.66	1.64	1.66	1.66	1.64	1.65	1.66
PMIS	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39
HMIS	1.60	1.60	1.59	1.59	1.59	1.59	1.59	1.58	1.58
PMIS-c1	1.59	1.58	1.58	1.59	1.58	1.58	1.59	1.58	1.58
PMIS-c2	1.32	1.32	1.32	1.32	1.32	1.32	1.32	1.32	1.32
CR-CLJP	2.44	2.44	2.44	2.44	2.44	2.44	2.44	2.44	2.44
CR-PMIS	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39

Table C.2: Grid complexities for the strongly scaled 7-point Laplacian.

p	1	2	4	8	16	32	64	128	256
Falgout	1.00	1.00	1.00	1.00	1.00	1.01	1.01	1.01	1.02
CLJP	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
CLJP-c	1.00	1.01	1.00	0.99	1.00	1.00	0.99	1.00	1.00
PMIS	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
HMIS	1.00	1.00	1.00	1.00	1.00	0.99	0.99	0.99	0.99
PMIS-c1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
PMIS-c2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
CR-CLJP	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
CR-PMIS	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table C.3: Grid complexities for the strongly scaled 7-point Laplacian relative to the single processor grid complexities.

p	1	2	4	8	16	32	64	128	256
Falgout	5.21	5.13	5.25	5.21	5.30	5.40	5.55	5.85	6.42
CLJP	27.95	27.72	27.67	27.92	27.95	27.70	27.88	27.78	27.88
CLJP-c	5.15	5.75	5.35	4.66	5.31	5.24	4.46	4.94	5.25
PMIS	2.36	2.36	2.36	2.36	2.37	2.36	2.37	2.36	2.37
HMIS	2.90	2.87	2.86	2.85	2.82	2.79	2.78	2.74	2.72
PMIS-c1	2.77	2.73	2.75	2.80	2.74	2.74	2.78	2.75	2.72
PMIS-c2	2.03	2.03	2.04	2.04	2.04	2.04	2.05	2.06	2.07
CR-CLJP	28.37	28.16	28.24	28.04	28.05	28.07	28.09	27.92	28.09
CR-PMIS	2.36	2.36	2.36	2.36	2.37	2.36	2.37	2.37	2.37

Table C.4: Operator complexities for the strongly scaled 7-point Laplacian.

p	1	2	4	8	16	32	64	128	256
Falgout	1.00	0.98	1.01	1.00	1.02	1.04	1.07	1.12	1.23
CLJP	1.00	0.99	0.99	1.00	1.00	0.99	1.00	0.99	1.00
CLJP-c	1.00	1.12	1.04	0.90	1.03	1.02	0.87	0.96	1.02
PMIS	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
HMIS	1.00	0.99	0.99	0.98	0.97	0.96	0.96	0.94	0.94
PMIS-c1	1.00	0.99	0.99	1.01	0.99	0.99	1.00	0.99	0.98
PMIS-c2	1.00	1.00	1.00	1.01	1.01	1.01	1.01	1.02	1.02
CR-CLJP	1.00	0.99	1.00	0.99	0.99	0.99	0.99	0.98	0.99
CR-PMIS	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table C.5: Operator complexities for the strongly scaled 7-point Laplacian relative to the single processor operator complexities.

p	1	2	4	8	16	32	64	128	256
Falgout	10.01	10.15	10.84	11.09	11.75	12.33	13.16	14.61	16.95
CLJP	92.34	91.97	93.60	95.77	97.26	96.90	96.44	100.71	101.60
CLJP-c	11.45	13.49	12.36	10.42	12.66	12.57	10.15	11.77	12.89
PMIS	49.71	49.80	52.27	52.63	53.33	54.41	52.92	51.78	53.43
HMIS	10.18	15.52	18.75	22.87	24.76	26.11	26.86	32.18	33.34
PMIS-c1	20.94	24.22	26.25	22.86	25.56	27.70	25.20	28.14	28.85
PMIS-c2	54.05	54.64	55.30	56.63	57.53	56.14	59.70	54.99	56.90
CR-CLJP	130.46	131.71	134.75	139.47	137.95	143.10	111.42	113.31	114.21
CR-PMIS	58.79	61.96	64.00	67.00	68.26	69.53	68.96	49.37	49.47

Table C.6: Amount of work per digit-of-accuracy for the strongly scaled 7-point Laplacian.

p	1	2	4	8	16	32	64	128	256
Falgout	1.00	1.01	1.08	1.11	1.17	1.23	1.31	1.46	1.69
CLJP	1.00	1.00	1.01	1.04	1.05	1.05	1.04	1.09	1.10
CLJP-c	1.00	1.18	1.08	0.91	1.11	1.10	0.89	1.03	1.13
PMIS	1.00	1.00	1.05	1.06	1.07	1.09	1.06	1.04	1.07
HMIS	1.00	1.52	1.84	2.25	2.43	2.56	2.64	3.16	3.27
PMIS-c1	1.00	1.16	1.25	1.09	1.22	1.32	1.20	1.34	1.38
PMIS-c2	1.00	1.01	1.02	1.05	1.06	1.04	1.10	1.02	1.05
CR-CLJP	1.00	1.01	1.03	1.07	1.06	1.10	0.85	0.87	0.88
CR-PMIS	1.00	1.05	1.09	1.14	1.16	1.18	1.17	0.84	0.84

Table C.7: Amount of work per digit-of-accuracy for the strongly scaled 7-point Laplacian relative to single processor WPDA.

p	1	2	4	8	16	32	64	128	256
Falgout	0.09	0.10	0.11	0.11	0.13	0.13	0.14	0.16	0.17
CLJP	0.25	0.25	0.26	0.26	0.27	0.27	0.26	0.28	0.28
CLJP-c	0.13	0.14	0.14	0.13	0.14	0.15	0.13	0.14	0.15
PMIS	0.80*	0.80*	0.81*	0.81*	0.82*	0.82*	0.81*	0.81*	0.82*
HMIS	0.27	0.43	0.49	0.56	0.59	0.61	0.62	0.68	0.69
PMIS-c1	0.54	0.60	0.62	0.57	0.61	0.63	0.60	0.64	0.65
PMIS-c2	0.84*	0.84*	0.84*	0.85*	0.85*	0.85*	0.85*	0.84*	0.85*
CR-CLJP	0.37	0.37	0.38	0.40	0.39	0.41	0.31	0.32	0.32
CR-PMIS	0.83*	0.84*	0.84*	0.85*	0.85*	0.86*	0.85*	0.80*	0.80*

Table C.8: Convergence factors for the strongly scaled 7-point Laplacian. Asterisks (*) denote trials that did not converge to a relative residual smaller than 10^{-8} within 100 iterations.

p	1	2	4	8	16	32	64	128	256
Falgout	91.01	98.74	53.06	25.73	16.71	11.50	9.54	9.44	12.01
CLJP	283.27	229.95	127.88	70.21	48.41	36.39	30.55	25.70	25.04
CLJP-c	94.86	107.32	53.39	23.25	16.12	10.61	6.03	6.27	7.59
PMIS	21.46	29.46	14.77	6.33	3.21	1.48	0.77	0.57	0.78
HMIS	43.11	67.94	33.10	13.52	6.71	2.81	1.24	0.72	0.84
PMIS-c1	39.88	65.96	33.07	13.95	7.00	3.15	1.53	0.96	0.91
PMIS-c2	24.57	24.80	12.78	5.77	2.95	1.50	0.87	0.71	0.82
CR-CLJP	433.54	304.93	167.82	90.04	58.59	42.26	33.93	26.98	25.60
CR-PMIS	40.36	38.99	19.77	8.78	4.41	2.02	0.99	0.75	0.95

Table C.9: Setup times in seconds for the strongly scaled 7-point Laplacian.

p	1	2	4	8	16	32	64	128	256
Falgout	1.00	1.08	0.58	0.28	0.18	0.13	0.10	0.10	0.13
CLJP	1.00	0.81	0.45	0.25	0.17	0.13	0.11	0.09	0.09
CLJP-c	1.00	1.13	0.56	0.25	0.17	0.11	0.06	0.07	0.08
PMIS	1.00	1.37	0.69	0.29	0.15	0.07	0.04	0.03	0.04
HMIS	1.00	1.58	0.77	0.31	0.16	0.07	0.03	0.02	0.02
PMIS-c1	1.00	1.65	0.83	0.35	0.18	0.08	0.04	0.02	0.02
PMIS-c2	1.00	1.01	0.52	0.24	0.12	0.06	0.04	0.03	0.03
CR-CLJP	1.00	0.70	0.39	0.21	0.14	0.10	0.08	0.06	0.06
CR-PMIS	1.00	0.97	0.49	0.22	0.11	0.05	0.02	0.02	0.02

Table C.10: Setup times for the strongly scaled 7-point Laplacian relative to single processor WPDA.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	2097152	2097152	2097152	2097152	2097152	2097152	2097152	2097152	2097152
2	1048576	1343226	1048576	647997	1048576	1048576	511604	1343226	647997
3	182810	756085	217481	136282	174784	156545	120968	756107	136279
4	56361	424617	64027	24123	22381	19363	23791	425141	24101
5	29029	228453	25695	3611	6070	2782	3673	229253	3602
6	13752	123782	10898	511	736	379	513	124361	495
7	6459	67704	4843	64	93	55	63	68435	70
8	3072	37022	2250	8	12	5	15	37888	
9	1408	19911	1087		1		3	20861	
10	629	10453	534					11156	
11	292	5206	238					5853	
12	123	2431	100					2857	
13	54	949	36					1222	
14	22	269	11						
15	9	53	4						
16		14							
17		1							

Table C.11: Number of degrees of freedom per level for the strongly scaled 7-point Laplacian on one processor.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	2097152	2097152	2097152	2097152	2097152	2097152	2097152	2097152	2097152
2	1052468	1343757	1048576	648126	1034570	1048576	519701	1343759	648126
3	225550	757291	215808	136769	149679	150813	121745	757300	136768
4	77081	425333	65355	24214	20070	17780	25573	425501	24203
5	33942	229231	25895	3629	2682	2443	3529	229461	3621
6	15183	124013	10758	504	362	324	478	124354	831
7	7104	67669	4665	65	53	40	65	68308	158
8	3441	36906	2137	14	11	8	7	37682	19
9	1752	19788	1043	3	1			20510	
10	926	10213	535					10816	
11	497	5071	276					5393	
12	237	2428	132					2439	
13	108	990	62					903	
14	48	312	21					223	
15	22	74	4						
16	13	15							
17	6	2							

Table C.12: Number of degrees of freedom per level for the strongly scaled 7-point Laplacian on 256 processors.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	14581760	14581760	14581760	14581760	14581760	14581760	14581760	14581760	14581760
2	19628800	21023896	19628800	11894693	19628800	19628800	8156004	21023896	11894693
3	6301804	42272007	9693681	6264778	5884574	4960421	5092566	42273233	6264725
4	7746095	57595959	8873563	1460375	1280389	1042075	1476657	57702135	1458735
5	8470117	60962649	8406311	204027	880410	160386	224025	61223337	204752
6	8252534	58084586	6210528	23897	75880	17283	24829	58247983	22687
7	5695185	52024374	4110467	1524	4759	1443	1623	52267103	2200
8	3418858	41034992	2292550	54	136	25	191	41422702	
9	1423860	28567265	971597		1		9	29640129	
10	378785	17710829	281974					18570792	
11	85032	9135100	56644					10523029	
12	15129	3638939	10000					4852021	
13	2916	800241	1296					1370874	
14	484	71379	121						
15	81	2809	16						
16		196							
17		1							

Table C.13: Number of nonzeros per level for the strongly scaled 7-point Laplacian on one processor.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	14581760	14581760	14581760	14581760	14581760	14581760	14581760	14581760	14581760
2	19650156	21024967	19628800	11901180	19333710	19628800	8443909	21024973	11901180
3	10058402	42354441	9410384	6303305	4642117	4598439	5267427	42355002	6303284
4	11080255	57524703	9163067	1468872	957926	765840	1682563	57539469	1468471
5	12029634	61143179	9032783	207375	116114	103491	206921	61173023	207181
6	10251259	58021773	6661838	22210	11864	11870	21708	58040064	65277
7	7675058	51617085	4285625	1817	1299	936	1759	52060888	8720
8	4706197	40583746	2372029	194	107	64	47	41462420	321
9	2402640	28453118	982431	9	1			29383762	
10	844228	17266671	285937					18082552	
11	247003	9149915	76176					9452991	
12	56169	3846134	17424					3596905	
13	11664	900270	3844					739041	
14	2304	96604	441					49335	
15	484	5476	16						
16	169	225							
17	36	4							

Table C.14: Number of nonzeros per level for the strongly scaled 7-point Laplacian on 256 processors.

C.2 Fixed-Size 3D Unstructured Laplacian

This section reports the results for experiments on a strongly scaled unstructured Laplacian problem. The continuous problem is the same as Section C.1, except now the Laplacian is discretized using finite elements on an unstructured mesh. The individual trials and problem size growth for each trial are listed in Table C.15. The problem contains approximately 940,000 degrees of freedom in all trials.

The data is organized into the following tables.

Trial information .... Table C.15

Grid complexities .... Table C.16

Relative grid complexities .... Table C.17

Operator complexities .... Table C.18

Relative operator complexities .... Table C.19

Amount of work per digit-of-accuracy .... Table C.20

Relative amount of work per digit-of-accuracy .... Table C.21

Convergence factors .... Table C.22

Setup times .... Table C.23

Relative setup times .... Table C.24

Level-by-level degrees of freedom .... Tables C.25 and C.26

Level-by-level nonzeros .... Tables C.27 and C.28

p	1	2	4	8	16	32	64	128	256	512
Relative n	1.00	1.11	1.11	1.11	1.11	1.11	1.11	1.11	1.11	1.11

Table C.15: Trials and relative trial sizes for the strongly scaled 3D unstructured Laplacian. The number of processors (p) is shown in the first row, and the number of unknowns (n) relative to the number of unknowns in the smallest trial is shown in the second row. This problem is strongly scaled, although the discretization package used marginally changed the problem size following the first trial.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.92	2.01	2.00	2.00	1.99	1.98	1.97	1.96	1.94	1.92
CLJP	1.70	1.79	1.79	1.80	1.80	1.79	1.79	1.80	1.79	1.80
CLJP-c	1.71	1.81	1.81	1.81	1.81	1.81	1.81	1.81	1.81	1.81
PMIS	1.23	1.23	1.23	1.23	1.23	1.23	1.23	1.23	1.23	1.23
HMIS	1.23	1.25	1.25	1.25	1.25	1.25	1.25	1.25	1.25	1.25
PMIS-c1	1.23	1.23	1.23	1.23	1.23	1.23	1.23	1.23	1.23	1.23
PMIS-c2	1.23	1.23	1.23	1.23	1.23	1.23	1.23	1.23	1.23	1.23
CR-CLJP	1.70	1.80	1.80	1.80	1.80	1.80	1.80	1.80	1.80	1.80
CR-PMIS	1.23	1.23	1.23	1.23	1.23	1.23	1.23	1.23	1.23	1.24

Table C.16: Grid complexities for the strongly scaled 3D unstructured Laplacian.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.00	1.05	1.04	1.04	1.04	1.03	1.03	1.02	1.01	1.00
CLJP	1.00	1.05	1.05	1.05	1.05	1.05	1.05	1.05	1.05	1.05
CLJP-c	1.00	1.05	1.06	1.06	1.06	1.06	1.06	1.06	1.06	1.06
PMIS	1.00	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01
HMIS	1.00	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.02	1.02
PMIS-c1	1.00	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01
PMIS-c2	1.00	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01
CR-CLJP	1.00	1.05	1.05	1.05	1.05	1.05	1.05	1.05	1.05	1.05
CR-PMIS	1.00	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01

Table C.17: Grid complexities for the strongly scaled 3D unstructured Laplacian relative to the single processor grid complexities.

p	1	2	4	8	16	32	64	128	256	512
Falgout	6.56	7.97	8.09	8.08	8.23	8.29	8.28	8.35	8.36	8.29
CLJP	5.05	6.71	6.71	6.72	6.75	6.70	6.73	6.73	6.73	6.76
CLJP-c	5.26	7.08	7.08	7.11	7.11	7.13	7.13	7.14	7.24	7.27
PMIS	1.46	1.47	1.47	1.47	1.47	1.47	1.47	1.47	1.47	1.47
HMIS	1.48	1.53	1.53	1.53	1.54	1.54	1.54	1.54	1.55	1.55
PMIS-c1	1.46	1.47	1.47	1.47	1.47	1.47	1.47	1.47	1.47	1.47
PMIS-c2	1.46	1.47	1.47	1.47	1.47	1.47	1.47	1.47	1.47	1.47
CR-CLJP	5.09	6.83	6.80	6.80	6.77	6.78	6.80	6.73	6.76	6.79
CR-PMIS	1.46	1.47	1.47	1.47	1.47	1.47	1.47	1.47	1.47	1.51

Table C.18: Operator complexities for the strongly scaled 3D unstructured Laplacian.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.00	1.22	1.23	1.23	1.25	1.26	1.26	1.27	1.27	1.26
CLJP	1.00	1.33	1.33	1.33	1.34	1.33	1.33	1.33	1.33	1.34
CLJP-c	1.00	1.35	1.35	1.35	1.35	1.35	1.36	1.36	1.38	1.38
PMIS	1.00	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01
HMIS	1.00	1.04	1.04	1.04	1.04	1.04	1.04	1.04	1.05	1.05
PMIS-c1	1.00	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.00	1.00
PMIS-c2	1.00	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01
CR-CLJP	1.00	1.34	1.34	1.34	1.33	1.33	1.34	1.32	1.33	1.33
CR-PMIS	1.00	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.04

Table C.19: Operator complexities for the strongly scaled 3D unstructured Laplacian relative to the single processor operator complexities.

p	1	2	4	8	16	32	64	128	256	512
Falgout	22.40	26.63	27.50	27.90	28.61	29.95	30.10	30.80	31.42	31.71
CLJP	18.17	23.32	23.36	23.99	24.36	24.48	24.89	24.97	25.13	25.52
CLJP-c	19.14	25.03	25.25	25.56	25.78	26.22	26.54	26.75	27.51	27.93
PMIS	26.22	26.82	28.15	27.11	27.39	28.50	27.00	27.82	29.39	29.19
HMIS	23.44	24.02	24.46	24.89	25.88	26.98	27.20	27.95	27.48	29.57
PMIS-c1	26.63	26.05	28.26	26.64	27.53	27.07	27.93	27.66	27.83	29.22
PMIS-c2	26.06	26.56	26.34	28.19	27.69	27.54	27.73	28.52	28.58	28.88
CR-CLJP	18.22	23.54	23.41	23.50	23.50	24.26	24.81	24.80	25.00	25.65
CR-PMIS	26.60	25.80	25.98	27.95	28.18	27.35	27.06	21.81	22.81	19.27

Table C.20: Amount of work per digit-of-accuracy for the strongly scaled 3D unstructured Laplacian.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.00	1.19	1.23	1.25	1.28	1.34	1.34	1.38	1.40	1.42
CLJP	1.00	1.28	1.29	1.32	1.34	1.35	1.37	1.37	1.38	1.40
CLJP-c	1.00	1.31	1.32	1.34	1.35	1.37	1.39	1.40	1.44	1.46
PMIS	1.00	1.02	1.07	1.03	1.04	1.09	1.03	1.06	1.12	1.11
HMIS	1.00	1.02	1.04	1.06	1.10	1.15	1.16	1.19	1.17	1.26
PMIS-c1	1.00	0.98	1.06	1.00	1.03	1.02	1.05	1.04	1.04	1.10
PMIS-c2	1.00	1.02	1.01	1.08	1.06	1.06	1.06	1.09	1.10	1.11
CR-CLJP	1.00	1.29	1.29	1.29	1.29	1.33	1.36	1.36	1.37	1.41
CR-PMIS	1.00	0.97	0.98	1.05	1.06	1.03	1.02	0.82	0.86	0.72

Table C.21: Amount of work per digit-of-accuracy for the strongly scaled 3D unstructured Laplacian relative to single processor WPDA.

p	1	2	4	8	16	32	64	128	256	512
Falgout	0.26	0.25	0.26	0.26	0.27	0.28	0.28	0.29	0.29	0.30
CLJP	0.28	0.27	0.27	0.28	0.28	0.28	0.29	0.29	0.29	0.30
CLJP-c	0.28	0.27	0.27	0.28	0.28	0.29	0.29	0.29	0.30	0.30
PMIS	0.77	0.78	0.79	0.78	0.78	0.79	0.78	0.78	0.79	0.79
HMIS	0.75	0.75	0.75	0.75	0.76	0.77	0.77	0.78	0.77	0.79
PMIS-c1	0.78	0.77	0.79	0.78	0.78	0.78	0.78	0.78	0.78	0.79
PMIS-c2	0.77	0.78	0.77	0.79	0.78	0.78	0.78	0.79	0.79	0.79
CR-CLJP	0.28	0.26	0.26	0.26	0.27	0.28	0.28	0.29	0.29	0.30
CR-PMIS	0.78	0.77	0.77	0.79	0.79	0.78	0.78	0.73	0.74	0.70

Table C.22: Convergence factors for the strongly scaled 3D unstructured Laplacian.

p	1	2	4	8	16	32	64	128	256	512
Falgout	56.74	63.20	35.21	18.57	14.37	10.42	7.02	5.91	7.33	8.38
CLJP	50.18	55.42	30.13	15.76	11.40	7.71	5.31	4.25	5.20	8.88
CLJP-c	54.60	58.84	32.25	17.10	12.07	8.24	5.98	4.86	5.28	7.34
PMIS	13.58	11.81	5.82	2.57	1.31	0.65	0.34	0.25	0.24	0.28
HMIS	14.92	12.85	6.33	2.80	1.43	0.70	0.38	0.27	0.26	0.36
PMIS-c1	16.13	12.96	6.46	2.90	1.52	0.78	0.47	0.34	0.34	0.57
PMIS-c2	17.97	13.89	6.77	3.04	1.59	0.83	0.49	0.35	0.34	0.36
CR-CLJP	79.49	76.66	40.49	20.54	13.62	8.93	5.76	4.47	6.19	12.70
CR-PMIS	24.57	18.17	8.66	3.84	1.91	0.93	0.47	0.38	0.37	0.54

Table C.23: Setup times in seconds for the strongly scaled 3D unstructured Laplacian.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.00	1.11	0.62	0.33	0.25	0.18	0.12	0.10	0.13	0.15
CLJP	1.00	1.10	0.60	0.31	0.23	0.15	0.11	0.08	0.10	0.18
CLJP-c	1.00	1.08	0.59	0.31	0.22	0.15	0.11	0.09	0.10	0.13
PMIS	1.00	0.87	0.43	0.19	0.10	0.05	0.03	0.02	0.02	0.02
HMIS	1.00	0.86	0.42	0.19	0.10	0.05	0.03	0.02	0.02	0.02
PMIS-c1	1.00	0.80	0.40	0.18	0.09	0.05	0.03	0.02	0.02	0.04
PMIS-c2	1.00	0.77	0.38	0.17	0.09	0.05	0.03	0.02	0.02	0.02
CR-CLJP	1.00	0.96	0.51	0.26	0.17	0.11	0.07	0.06	0.08	0.16
CR-PMIS	1.00	0.74	0.35	0.16	0.08	0.04	0.02	0.02	0.02	0.02

Table C.24: Setup times for the strongly scaled 3D unstructured Laplacian relative to single processor setup times.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	939488	939488	939488	939488	939488	939488	939488	939488	939488
2	429340	368065	371082	176299	177667	175732	175572	368075	176296
3	217854	158634	160172	30611	33034	31150	30486	158714	30673
4	110045	72139	73323	4831	5355	4945	4803	72184	4826
5	55167	33422	34614	698	846	718	731	33658	699
6	27270	15659	16516	112	122	105	96	15915	87
7	13080	7100	7622	19	15	18	12	7466	
8	6096	3118	3360	2	5	2	1	3439	
9	2689	1334	1459					1585	
10	1137	519	600					706	
11	464	205	233						
12	165	75	87						
13	52	15	28						
14	10	2	6						
15	2								

Table C.25: Number of degrees of freedom per level for the strongly scaled 3D unstructured Laplacian on one processor.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	1046902	1046902	1046902	1046902	1046902	1046902	1046902	1046902	1046902
2	467284	429152	431626	203670	216360	203417	203337	429165	203605
3	240287	203231	205977	34600	38422	34700	34662	203309	34573
4	124886	100339	102944	5440	6138	5345	5432	100420	9512
5	65278	50560	53074	828	944	822	816	50853	1765
6	34194	25820	27977	127	138	121	112	25944	403
7	17795	12952	14544	15	18	18	15	13081	65
8	8978	6190	7232	2	3	3	2	6343	
9	4422	2760	3390					2901	
10	2004	1147	1512					1202	
11	876	463	650					458	
12	372	175	266					160	
13	140	62	104						
14	50	18	34						
15	16	6	12						
16	3		4						

Table C.26: Number of degrees of freedom per level for the strongly scaled 3D unstructured Laplacian on 512 processors.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	12293914	12293914	12293914	12293914	12293914	12293914	12293914	12293914	12293914
2	11997994	11632525	12154984	4177823	4150047	4160304	4144170	11632709	4177784
3	14350484	11616612	11923126	1199825	1380250	1243344	1189412	11615462	1205573
4	12879031	9190163	9588573	230289	275095	240071	227055	9184540	230528
5	10465763	7029110	7443920	29694	40938	31174	32387	6995362	31073
6	7830564	4892689	5292520	3716	4128	3395	2872	4850721	2905
7	5234182	2964700	3241658	283	185	252	136	3058688	
8	3155192	1543014	1694030	4	25	4	1	1669245	
9	1578069	682064	764119					876473	
10	640513	204229	257202					352518	
11	182010	41231	53109						
12	26859	5623	7567						
13	2704	225	784						
14	100	4	36						
15	4								

Table C.27: Number of nonzeros per level for the strongly scaled 3D unstructured Laplacian on one processor.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	13788698	13788698	13788698	13788698	13788698	13788698	13788698	13788698	13788698
2	13781152	13035890	13411986	4823226	5562126	4828127	4819333	13036227	4822215
3	17755237	15582507	16117627	1336574	1622894	1346072	1340566	15586963	1336267
4	17853496	14569153	15257226	252058	306200	247885	252194	14577442	755546
5	15798244	12295910	13276744	36274	43210	36746	36436	12361421	134369
6	13126460	9900110	11040579	4165	4660	3847	3440	9864674	35755
7	9928865	6959986	8128122	167	296	224	177	6977131	2869
8	6427188	4145334	5062152	4	9	9	4	4262753	
9	3560532	1991124	2614646					2109967	
10	1574630	749591	1104128					785580	
11	546360	194071	345800					188664	
12	132120	30593	70018					25572	
13	19488	3844	10816						
14	2500	324	1156						
15	256	36	144						
16	9		16						

Table C.28: Number of nonzeros per level for the strongly scaled 3D unstructured Laplacian on 512 processors.

C.3 Scaled 3D 7-Point Laplacian

This section reports the results for experiments on a weakly scaled 7-point Laplacian problem. The continuous problem is the same as Section C.1, except now the problem is scaled to assign the same number of unknowns to each processor for all trials. On one processor, the problem contains 125,000 unknowns. On 256 processors, the problem contains 32 million unknowns. The individual trials and problem size growth for each trial are listed in Table C.29.

The data is organized into the following tables.

Trial information .... Table C.29
Grid complexities .... Table C.30
Relative grid complexities .... Table C.31
Operator complexities .... Table C.32
Relative operator complexities .... Table C.33
Amount of work per digit-of-accuracy .... Table C.34
Relative amount of work per digit-of-accuracy .... Table C.35
Convergence factors .... Table C.36
Setup times .... Table C.37
Relative setup times .... Table C.38
Level-by-level degrees of freedom .... Tables C.39 and C.40
Level-by-level nonzeros .... Tables C.41 and C.42

p	1	2	4	8	16	32	64	256	512
Relative n	1	2	4	8	16	32	64	256	512

Table C.29: Trials and relative trial sizes for the weakly scaled 3D 7-point Laplacian. The number of processors (p) is shown in the first row, and the number of unknowns (n) relative to the number of unknowns in the smallest trial is shown in the second row.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.65	1.65	1.65	1.65	1.65	1.65	1.65	–	1.65	–
CLJP	2.40	2.41	2.42	2.43	2.44	2.44	2.45	–	2.45	2.45
CLJP-c	1.64	1.64	1.64	1.64	1.64	1.64	1.64	–	1.64	1.64
PMIS	1.40	1.39	1.39	1.39	1.39	1.39	1.39	–	1.39	1.38
HMIS	1.60	1.60	1.59	1.59	1.59	1.59	1.59	–	1.59	1.59
PMIS-c1	1.59	1.59	1.59	1.59	1.59	1.59	1.59	–	1.59	1.58
PMIS-c2	1.33	1.33	1.32	1.32	1.32	1.32	1.31	–	1.31	1.31
CR-CLJP	2.40	2.42	2.43	2.44	2.44	2.44	2.45	–	2.45	2.46
CR-PMIS	1.40	1.39	1.39	1.39	1.39	1.39	1.39	–	1.39	1.39

Table C.30: Grid complexities for the weakly scaled 3D 7-point Laplacian.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.00	1.00	1.00	1.00	1.00	1.00	1.00	–	1.00	–
CLJP	1.00	1.01	1.01	1.02	1.02	1.02	1.02	–	1.02	1.02
CLJP-c	1.00	1.00	1.00	1.00	1.00	1.00	1.00	–	1.00	1.00
PMIS	1.00	1.00	1.00	1.00	0.99	0.99	0.99	–	0.99	0.99
HMIS	1.00	1.00	1.00	1.00	1.00	1.00	0.99	–	0.99	0.99
PMIS-c1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	–	1.00	1.00
PMIS-c2	1.00	1.00	1.00	0.99	0.99	0.99	0.99	–	0.99	0.99
CR-CLJP	1.00	1.00	1.01	1.01	1.02	1.02	1.02	–	1.02	1.02
CR-PMIS	1.00	1.00	1.00	1.00	0.99	0.99	0.99	–	0.99	0.99

Table C.31: Grid complexities for the weakly scaled 3D 7-point Laplacian relative to the single processor grid complexities.

p	1	2	4	8	16	32	64	128	256	512
Falgout	4.18	4.38	4.70	5.04	5.25	5.57	6.02	–	6.69	–
CLJP	19.83	21.56	23.82	26.32	27.41	28.76	30.33	–	32.13	33.05
CLJP-c	3.88	4.00	4.16	4.40	4.55	4.73	4.95	–	5.30	5.53
PMIS	2.32	2.34	2.35	2.36	2.36	2.37	2.37	–	2.38	2.38
HMIS	2.82	2.81	2.82	2.82	2.83	2.82	2.83	–	2.83	2.83
PMIS-c1	2.77	2.77	2.78	2.78	2.79	2.79	2.79	–	2.79	2.79
PMIS-c2	2.04	2.05	2.04	2.04	2.04	2.05	2.05	–	2.05	2.05
CR-CLJP	20.77	22.17	24.11	26.70	27.78	28.91	30.48	–	32.21	33.13
CR-PMIS	2.32	2.34	2.35	2.36	2.36	2.37	2.37	–	2.38	2.38

Table C.32: Operator complexities for the weakly scaled 3D 7-point Laplacian.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.00	1.05	1.13	1.21	1.26	1.33	1.44	–	1.60	–
CLJP	1.00	1.09	1.20	1.33	1.38	1.45	1.53	–	1.62	1.67
CLJP-c	1.00	1.03	1.07	1.13	1.17	1.22	1.28	–	1.37	1.43
PMIS	1.00	1.01	1.01	1.02	1.02	1.02	1.02	–	1.02	1.02
HMIS	1.00	1.00	1.00	1.00	1.00	1.00	1.00	–	1.00	1.00
PMIS-c1	1.00	1.00	1.00	1.00	1.00	1.01	1.01	–	1.01	1.01
PMIS-c2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	–	1.00	1.00
CR-CLJP	1.00	1.07	1.16	1.29	1.34	1.39	1.47	–	1.55	1.59
CR-PMIS	1.00	1.01	1.01	1.02	1.02	1.02	1.02	–	1.02	1.03

Table C.33: Operator complexities for the weakly scaled 3D 7-point Laplacian relative to the single processor operator complexities.

p	1	2	4	8	16	32	64	128	256	512
Falgout	6.15	7.01	8.44	10.20	11.25	13.05	16.06	–	21.27	–
CLJP	49.51	57.36	69.00	84.36	93.74	102.99	116.55	–	137.60	154.53
CLJP-c	6.00	6.53	7.52	8.98	9.99	11.20	13.12	–	16.42	19.35
PMIS	26.23	27.70	34.03	44.06	47.40	54.30	69.19	–	83.90	104.58
HMIS	5.35	11.47	15.13	20.04	21.15	24.19	32.79	–	41.85	60.12
PMIS-c1	10.23	12.36	16.12	21.12	21.31	24.26	35.00	–	41.48	56.58
PMIS-c2	30.53	32.08	36.39	47.31	51.73	58.78	72.63	–	87.92	108.02
CR-CLJP	74.59	57.26	68.58	85.47	91.17	104.01	117.27	–	136.43	154.46
CR-PMIS	29.05	27.61	31.72	45.01	46.76	52.62	69.18	–	75.09	80.39

Table C.34: Amount of work per digit-of-accuracy for the weakly scaled 3D 7-point Laplacian.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.00	1.14	1.37	1.66	1.83	2.12	2.61	–	3.46	–
CLJP	1.00	1.16	1.39	1.70	1.89	2.08	2.35	–	2.78	3.12
CLJP-c	1.00	1.09	1.25	1.50	1.67	1.87	2.19	–	2.74	3.23
PMIS	1.00	1.06	1.30	1.68	1.81	2.07	2.64	–	3.20	3.99
HMIS	1.00	2.15	2.83	3.75	3.95	4.52	6.13	–	7.82	11.24
PMIS-c1	1.00	1.21	1.58	2.06	2.08	2.37	3.42	–	4.05	5.53
PMIS-c2	1.00	1.05	1.19	1.55	1.69	1.93	2.38	–	2.88	3.54
CR-CLJP	1.00	0.77	0.92	1.15	1.22	1.39	1.57	–	1.83	2.07
CR-PMIS	1.00	0.95	1.09	1.55	1.61	1.81	2.38	–	2.59	2.77

Table C.35: Amount of work per digit-of-accuracy for the weakly scaled 3D 7-point Laplacian relative to single processor WPDA.

p	1	2	4	8	16	32	64	128	256	512
Falgout	0.04	0.06	0.08	0.10	0.12	0.14	0.18	–	0.23	–
CLJP	0.16	0.18	0.20	0.24	0.26	0.28	0.30	–	0.34	0.37
CLJP-c	0.05	0.06	0.08	0.10	0.12	0.14	0.18	–	0.23	0.27
PMIS	0.66	0.68	0.73	0.78	0.79	0.82*	0.85*	–	0.88*	0.90*
HMIS	0.09	0.32	0.42	0.52	0.54	0.58	0.67	–	0.73	0.81
PMIS-c1	0.29	0.36	0.45	0.55	0.55	0.59	0.69	–	0.73	0.80
PMIS-c2	0.74	0.75	0.77	0.82*	0.83*	0.85*	0.88*	–	0.90*	0.92*
CR-CLJP	0.28	0.17	0.20	0.24	0.25	0.28	0.30	–	0.34	0.37
CR-PMIS	0.69	0.68	0.71	0.79	0.79	0.81*	0.85*	–	0.86*	0.87*

Table C.36: Convergence factors for the weakly scaled 3D 7-point Laplacian. Asterisks (*) denote trials that did not converge to a relative residual smaller than 10^{-8} within 100 iterations.

p	1	2	4	8	16	32	64	128	256	512
Falgout	3.17	4.68	7.19	9.95	13.18	20.24	34.51	–	98.19	–
CLJP	9.22	13.53	19.25	28.79	38.74	54.51	81.18	–	110.13	143.47
CLJP-c	3.21	4.68	6.28	8.63	11.21	15.59	23.98	–	49.65	112.19
PMIS	0.96	1.46	1.89	2.29	2.69	3.08	3.51	–	3.63	3.78
HMIS	1.81	2.82	3.72	4.64	5.50	6.35	7.22	–	7.32	7.46
PMIS-c1	1.86	2.89	3.83	4.81	5.83	6.98	8.36	–	11.30	15.15
PMIS-c2	1.11	1.49	1.81	2.17	2.53	3.00	3.66	–	6.09	9.35
CR-CLJP	15.62	20.32	26.42	37.60	48.05	65.30	94.76	–	124.20	157.94
CR-PMIS	1.96	2.50	2.89	3.35	3.77	4.21	4.63	–	4.93	5.14

Table C.37: Setup times in seconds for the weakly scaled 3D 7-point Laplacian.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.00	1.48	2.27	3.14	4.16	6.39	10.90	–	31.00	–
CLJP	1.00	1.47	2.09	3.12	4.20	5.91	8.80	–	11.94	15.55
CLJP-c	1.00	1.46	1.96	2.69	3.50	4.86	7.48	–	15.49	34.99
PMIS	1.00	1.51	1.96	2.38	2.79	3.20	3.64	–	3.77	3.93
HMIS	1.00	1.55	2.05	2.56	3.03	3.50	3.98	–	4.04	4.11
PMIS-c1	1.00	1.55	2.06	2.58	3.13	3.75	4.49	–	6.07	8.14
PMIS-c2	1.00	1.34	1.63	1.96	2.29	2.71	3.30	–	5.50	8.44
CR-CLJP	1.00	1.30	1.69	2.41	3.08	4.18	6.07	–	7.95	10.11
CR-PMIS	1.00	1.27	1.47	1.71	1.92	2.14	2.36	–	2.51	2.62

Table C.38: Setup times for the weakly scaled 3D 7-point Laplacian relative to single processor setup times.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	125000	125000	125000	125000	125000	125000	125000	125000	125000
2	62500	79255	62500	39654	62500	62500	31898	79257	39652
3	11612	43927	12346	8140	10425	9925	7560	43960	8134
4	3676	24461	3488	1449	1398	1220	1444	24575	1429
5	1812	12962	1306	219	275	198	218	13044	224
6	814	6826	532	31	31	32	34	6985	24
7	354	3578	223	9	6	5	5	3923	
8	150	1794	89					2179	
9	64	885	35					1120	
10	21	433	9					510	
11	6	207							
12		85							
13		28							
14		7							

Table C.39: Number of degrees of freedom per level for the weakly scaled 7-point Laplacian on one processor.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	32000000	32000000	32000000	32000000	32000000	32000000	32000000	32000000	32000000
2	16009179	20537140	16000000	9826611	15966297	16000000	7742507	20537140	9826610
3	2992766	11602206	3074943	2077485	2462967	2404317	1853279	11602225	2077484
4	935783	6535071	832887	365214	318298	279978	360639	6535416	365194
5	443826	3538715	322364	54187	60017	38320	54477	3539296	54192
6	198984	1932744	136873	7290	7313	4888	7371	1933765	7307
7	92364	1071516	60559	923	733	672	951	1073374	1522
8	44121	595122	28640	115	92	87	137	597406	247
9	22556	326139	14281	21	9	12	21	328826	28
10	12695	174508	7461	4		2	4	176358	
11	7778	89753	4016					91046	
12	4941	42653	2237					43888	
13	3093	17053	1206					17939	
14	1901	4594	600					5331	
15	1152	806	245					1113	
16	693	143	103					271	
17	401	33	39						
18	218	6	5						
19	121								
20	60								
21	30								
22	14								
23	6								

Table C.40: Number of degrees of freedom per level for the weakly scaled 7-point Laplacian on 256 processors.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	860000	860000	860000	860000	860000	860000	860000	860000	860000
2	1142800	1223341	1142800	711008	1142800	1142800	505954	1223355	710980
3	389618	2364327	472242	344846	329803	315431	302412	2366882	344784
4	416134	3082833	383290	73159	69906	56970	74632	3085625	72423
5	394834	3037866	279348	8791	18873	8236	8608	3045586	9310
6	265424	2588574	144938	603	805	634	716	2632951	484
7	95900	1974206	45039	73	36	25	25	2158267	
8	21892	1157378	7921					1483151	
9	4086	537655	1225					761470	
10	441	178329	81					244852	
11	36	42839							
12		7225							
13		784							
14		49							

Table C.41: Number of nonzeros per level for the weakly scaled 7-point Laplacian on one processor.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	223360000	223360000	223360000	223360000	223360000	223360000	223360000	223360000	223360000
2	302133277	323427864	302082000	182093931	301369647	302082000	124904993	323427864	182093918
3	120014580	658873682	130660149	98174029	81861349	79547595	80955251	658874499	98174006
4	140263153	909933719	115717887	23344694	18814524	15354572	23865675	909981002	23344070
5	161142912	987862213	116313558	3449577	5735329	2279066	3729981	987997392	3450214
6	149658282	974291068	94101133	410896	438749	242440	439551	974915181	412387
7	119574240	914891164	73236057	42939	29753	26696	45599	916278888	122738
8	90654189	769372366	52906240	3497	2454	2271	4501	771122388	14353
9	67922418	582417167	35695421	319	75	116	289	585392578	648
10	50467635	403848726	21442425	16		4	16	405492180	
11	34688164	251540013	11041088					252915230	
12	20291569	128043921	4674717					131450602	
13	9323641	42265579	1450438					44703149	
14	3611389	5561302	359974					6906353	
15	1327096	297656	60025					496627	
16	480249	15315	10609					44983	
17	160801	1039	1521						
18	47524	36	25						
19	14641								
20	3600								
21	900								
22	196								
23	36								

Table C.42: Number of nonzeros per level for the weakly scaled 7-point Laplacian on 256 processors.

C.4 Scaled 3D Unstructured Laplacian

This section reports the results for experiments on a weakly scaled 3D unstructured Laplacian problem. The continuous problem is the same as Section C.2, except now the problem is scaled to assign approximately the same number of unknowns to each processor for all trials. On one processor the problem has approximately approximately 211,000 unknowns. On 512 processors there is approximately 100 million unknowns, giving an average of 198,000 unknowns per processor. The individual trials and problem size growth for each trial are listed in Table C.43.

The data is organized into the following tables.

Trial information .... Table C.43
Grid complexities .... Table C.44
Relative grid complexities .... Table C.45
Operator complexities .... Table C.46
Relative operator complexities .... Table C.47
Amount of work per digit-of-accuracy .... Table C.48
Relative amount of work per digit-of-accuracy .... Table C.49
Convergence factors .... Table C.50
Setup times .... Table C.51
Relative setup times .... Table C.52
Level-by-level degrees of freedom .... Tables C.53 and C.54
Level-by-level nonzeros .... Tables C.55 and C.56

p	1	2	4	8	16	32	64	128	256	512
Relative n	1.00	2.56	4.95	7.66	18.41	37.12	60.26	139.09	285.77	478.92

Table C.43: Trials and relative trial sizes for the weakly scaled 3D unstructured Laplacian. The number of processors (p) is shown in the first row, and the number of unknowns (n) relative to the number of unknowns in the smallest trial is shown in the second row.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.84	1.84	2.00	2.02	1.84	2.01	2.02	1.78	1.97	1.97
CLJP	1.67	1.65	1.79	1.82	1.68	1.82	1.88	1.68	1.82	1.91
CLJP-c	1.66	1.66	1.81	1.83	1.68	1.83	1.88	1.66	1.82	1.87
PMIS	1.24	1.21	1.23	1.24	1.21	1.27	1.25	1.21	1.29	1.26
HMIS	1.25	1.23	1.25	1.27	1.23	1.28	1.30	1.25	1.31	1.33
PMIS-c1	1.24	1.21	1.23	1.24	1.21	1.27	1.25	1.21	1.29	1.28
PMIS-c2	1.23	1.21	1.23	1.24	1.21	1.27	1.25	1.21	1.29	1.25
CR-CLJP	1.68	1.66	1.80	1.82	1.68	1.82	1.88	1.68	1.82	1.91
CR-PMIS	1.23	1.21	1.23	1.24	1.21	1.27	1.25	1.21	1.29	1.26

Table C.44: Grid complexities for the weakly scaled 3D unstructured Laplacian.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.00	1.00	1.09	1.09	1.00	1.09	1.10	0.97	1.07	1.07
CLJP	1.00	0.99	1.07	1.08	1.00	1.09	1.12	1.00	1.09	1.14
CLJP-c	1.00	1.00	1.09	1.10	1.01	1.10	1.13	1.00	1.09	1.12
PMIS	1.00	0.98	1.00	1.00	0.98	1.03	1.01	0.98	1.04	1.02
HMIS	1.00	0.99	1.00	1.01	0.99	1.03	1.04	1.00	1.05	1.06
PMIS-c1	1.00	0.98	1.00	1.00	0.98	1.02	1.01	0.98	1.04	1.03
PMIS-c2	1.00	0.98	1.00	1.00	0.98	1.02	1.01	0.98	1.04	1.01
CR-CLJP	1.00	0.99	1.07	1.08	1.00	1.09	1.12	1.00	1.08	1.14
CR-PMIS	1.00	0.98	1.00	1.00	0.98	1.03	1.01	0.98	1.04	1.02

Table C.45: Grid complexities for the weakly scaled 3D unstructured Laplacian relative to the single processor grid complexities.

p	1	2	4	8	16	32	64	128	256	512
Falgout	6.06	7.46	8.09	8.89	8.80	9.48	10.76	9.72	10.52	*
CLJP	4.70	5.71	6.71	7.01	6.77	7.77	8.58	7.63	8.70	9.72
CLJP-c	4.73	5.95	7.08	7.33	7.07	8.25	9.03	7.91	9.33	9.87
PMIS	1.47	1.52	1.47	1.47	1.46	1.53	1.49	1.44	1.60	1.52
HMIS	1.54	1.67	1.53	1.65	1.65	1.60	1.79	1.71	1.71	1.95
PMIS-c1	1.49	1.52	1.47	1.48	1.47	1.53	1.51	1.46	1.63	1.63
PMIS-c2	1.47	1.52	1.47	1.47	1.46	1.53	1.48	1.44	1.59	1.49
CR-CLJP	4.79	5.84	6.80	7.03	6.83	7.81	8.61	7.67	8.74	9.74
CR-PMIS	1.47	1.52	1.47	1.47	1.46	1.53	1.49	1.44	1.60	1.52

Table C.46: Operator complexities for the weakly scaled 3D unstructured Laplacian. The operator complexity for Falgout on 512 processors was corrupted due to overflow.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.00	1.23	1.33	1.47	1.45	1.56	1.78	1.60	1.74	*
CLJP	1.00	1.21	1.43	1.49	1.44	1.65	1.83	1.62	1.85	2.07
CLJP-c	1.00	1.26	1.50	1.55	1.49	1.74	1.91	1.67	1.97	2.09
PMIS	1.00	1.03	1.00	1.00	0.99	1.04	1.01	0.98	1.09	1.03
HMIS	1.00	1.09	1.00	1.07	1.07	1.04	1.16	1.11	1.12	1.27
PMIS-c1	1.00	1.02	0.99	0.99	0.99	1.03	1.02	0.98	1.10	1.09
PMIS-c2	1.00	1.03	1.00	1.00	0.99	1.04	1.00	0.97	1.08	1.01
CR-CLJP	1.00	1.22	1.42	1.47	1.43	1.63	1.80	1.60	1.83	2.03
CR-PMIS	1.00	1.03	1.00	1.00	0.99	1.04	1.01	0.98	1.09	1.03

Table C.47: Operator complexities for the weakly scaled 3D unstructured Laplacian relative to the single processor operator complexities. The operator complexity for Falgout on 512 processors was corrupted due to overflow.

p	1	2	4	8	16	32	64	128	256	512
Falgout	18.35	23.19	27.50	32.30	34.57	37.81	47.66	45.78	50.76	*
CLJP	14.19	17.78	23.36	25.48	25.89	30.82	36.22	34.19	38.81	47.18
CLJP-c	14.40	18.63	25.25	26.70	27.54	33.25	38.14	36.08	42.77	48.42
PMIS	17.81	24.59	28.15	30.77	41.24	41.61	44.37	51.80	68.05	67.59
HMIS	16.45	22.30	24.46	30.52	38.68	38.93	50.52	54.77	57.49	73.98
PMIS-c1	18.38	25.87	28.26	31.71	37.59	42.18	46.12	51.39	69.08	66.89
PMIS-c2	17.91	23.66	26.34	30.70	38.93	42.23	44.35	51.77	69.56	65.49
CR-CLJP	14.80	18.20	23.41	25.24	26.29	30.63	35.22	33.98	39.55	45.22
CR-PMIS	17.42	24.13	25.98	29.77	39.16	42.88	43.72	46.55	46.96	45.02

Table C.48: Amount of work per digit-of-accuracy for the weakly scaled 3D unstructured Laplacian. Overflow corrupted the results for Falgout on 512 processors.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.00	1.26	1.50	1.76	1.88	2.06	2.60	2.50	2.77	*
CLJP	1.00	1.25	1.65	1.80	1.82	2.17	2.55	2.41	2.73	3.32
CLJP-c	1.00	1.29	1.75	1.85	1.91	2.31	2.65	2.50	2.97	3.36
PMIS	1.00	1.38	1.58	1.73	2.32	2.34	2.49	2.91	3.82	3.79
HMIS	1.00	1.36	1.49	1.86	2.35	2.37	3.07	3.33	3.49	4.50
PMIS-c1	1.00	1.41	1.54	1.72	2.05	2.29	2.51	2.80	3.76	3.64
PMIS-c2	1.00	1.32	1.47	1.71	2.17	2.36	2.48	2.89	3.88	3.66
CR-CLJP	1.00	1.23	1.58	1.71	1.78	2.07	2.38	2.30	2.67	3.06
CR-PMIS	1.00	1.38	1.49	1.71	2.25	2.46	2.51	2.67	2.70	2.58

Table C.49: Amount of work per digit-of-accuracy for the weakly scaled 3D unstructured Laplacian relative to single processor WPDA. Overflow corrupted the results for Falgout on 512 processors.

p	1	2	4	8	16	32	64	128	256	512
Falgout	0.22	0.23	0.26	0.28	0.31	0.32	0.35	0.38	0.39	0.41
CLJP	0.22	0.23	0.27	0.28	0.30	0.31	0.34	0.36	0.36	0.39
CLJP-c	0.22	0.23	0.27	0.28	0.31	0.32	0.34	0.36	0.37	0.39
PMIS	0.68	0.75	0.79	0.80	0.85*	0.84*	0.86*	0.88*	0.90*	0.90*
HMIS	0.65	0.71	0.75	0.78	0.82	0.83	0.85*	0.87*	0.87*	0.89*
PMIS-c1	0.69	0.76	0.79	0.81	0.84	0.85*	0.86*	0.88*	0.90*	0.89*
PMIS-c2	0.68	0.74	0.77	0.80	0.84*	0.85*	0.86*	0.88*	0.90*	0.90*
CR-CLJP	0.23	0.23	0.26	0.28	0.30	0.31	0.32	0.35	0.36	0.37
CR-PMIS	0.68	0.75	0.77	0.80	0.84*	0.85*	0.86*	0.87*	0.85*	0.86*

Table C.50: Convergence factors for the weakly scaled 3D unstructured Laplacian. Asterisks (*) denote trials that did not converge to a relative residual smaller than 10^{-8} within 100 iterations.

p	1	2	4	8	16	32	64	128	256	512
Falgout	9.46	25.16	35.14	36.00	63.73	88.95	100.92	137.89	187.93	267.50
CLJP	8.28	21.36	30.28	29.03	49.13	71.09	77.48	95.04	131.98	178.87
CLJP-c	8.87	22.49	32.29	31.35	52.15	79.39	88.46	112.90	162.60	208.55
PMIS	2.41	5.44	5.82	4.65	6.66	8.89	6.48	7.15	11.92	8.18
HMIS	2.72	6.14	6.30	5.70	8.19	9.28	9.19	8.83	11.92	13.66
PMIS-c1	2.81	6.01	6.42	5.16	7.55	10.14	8.40	10.38	18.34	20.09
PMIS-c2	3.03	6.40	6.77	5.38	7.93	10.40	8.11	10.47	18.35	17.61
CR-CLJP	13.42	30.14	40.20	37.73	59.86	85.06	89.81	109.22	151.11	191.76
CR-PMIS	4.15	8.37	8.63	6.92	9.46	11.84	9.00	9.93	15.11	11.69

Table C.51: Setup times in seconds for the weakly scaled 3D unstructured Laplacian.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.00	2.66	3.72	3.81	6.74	9.41	10.67	14.58	19.87	28.29
CLJP	1.00	2.58	3.66	3.51	5.94	8.59	9.36	11.49	15.95	21.61
CLJP-c	1.00	2.54	3.64	3.54	5.88	8.95	9.97	12.73	18.33	23.51
PMIS	1.00	2.26	2.42	1.93	2.77	3.69	2.69	2.97	4.96	3.40
HMIS	1.00	2.26	2.32	2.10	3.01	3.41	3.38	3.25	4.39	5.03
PMIS-c1	1.00	2.14	2.28	1.83	2.68	3.60	2.98	3.69	6.52	7.14
PMIS-c2	1.00	2.11	2.23	1.77	2.61	3.43	2.68	3.45	6.05	5.81
CR-CLJP	1.00	2.25	3.00	2.81	4.46	6.34	6.69	8.14	11.26	14.29
CR-PMIS	1.00	2.02	2.08	1.67	2.28	2.85	2.17	2.39	3.64	2.81

Table C.52: Setup times for the weakly scaled 3D unstructured Laplacian relative to single processor setup times.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	211369	211369	211369	211369	211369	211369	211369	211369	211369
2	88030	79398	78489	41201	43321	41812	41133	79410	41201
3	44745	33924	33131	7110	7784	7062	7095	33982	7085
4	23166	15414	14992	1176	1365	1193	1161	15586	1154
5	11816	7322	7185	178	226	197	181	7507	165
6	5858	3490	3429	27	33	29	30	3653	
7	2852	1607	1643	4	6	2	6	1811	
8	1291	708	762					840	
9	547	295	321					349	
10	205	119	111						
11	68	43	36						
12	19	12	10						
13	4	2	3						

Table C.53: Number of degrees of freedom per level for the weakly scaled 3D unstructured Laplacian on one processor.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	60403737	60403737	60403737	60403737	60403737	60403737	60403737	60403737	60403737
2	27671439	25285687	25318470	14359744	15139077	14542218	14245638	25285718	14359656
3	14373376	11879884	11809449	2607767	3071949	2678195	2584476	11880323	2607418
4	7731487	5948960	5901896	403990	494719	412398	399734	5948970	403722
5	4170398	3050560	3061373	58696	73373	59584	58261	3051512	58783
6	2236836	1591203	1624733	8030	10316	8129	7901	1593456	14219
7	1189293	834851	870053	1060	1363	1034	1044	836847	2428
8	624616	434550	463753	140	168	149	137	436687	492
9	324079	221914	243203	23	22	22	23	223947	78
10	166309	110007	124764	2	4	3	2	111618	
11	84564	52084	61732					53769	
12	41960	23029	29041					24345	
13	19644	8908	12535					9902	
14	7798	2727	4550					3273	
15	2179	644	1218					824	
16	422	124	246					216	
17	82	24	55						
18	21	1	15						
19	6		2						

Table C.54: Number of degrees of freedom per level for the weakly scaled 3D unstructured Laplacian on 256 processors.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	2763227	2763227	2763227	2763227	2763227	2763227	2763227	2763227	2763227
2	2432480	2294810	2296439	972697	1084093	1021862	978505	2294902	972697
3	2845345	2313826	2303497	273174	321466	270006	272831	2314544	272707
4	2830398	2008466	1982092	49542	64761	50713	48307	2025910	49238
5	2372354	1550988	1565975	6354	9056	6951	6009	1581167	6283
6	1678858	1082628	1078633	475	729	569	638	1114673	
7	1053898	609345	663311	16	36	4	36	688369	
8	536529	267570	316342					336524	
9	191307	77197	90649					105235	
10	39217	14111	12185						
11	4610	1849	1296						
12	361	144	100						
13	16	4	9						

Table C.55: Number of nonzeros per level for the weakly scaled 3D unstructured Laplacian on one processor.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	808335181	808335181	808335181	808335181	808335181	808335181	808335181	808335181	808335181
2	754051399	782569055	820264568	347014194	390319739	363892034	341902116	782569616	347012600
3	1008280890	937920562	972922855	112833583	149771233	119765723	111441264	937971509	112812690
4	1101287779	933539964	969328320	22047158	31081925	22899112	21726950	933483126	22038994
5	1075171794	866975624	912621829	3352940	4743931	3418138	3323103	867410732	3358631
6	974978232	774456705	831365727	439090	610052	442871	430131	775823930	1359743
7	835036565	649378457	709486527	51080	66685	48322	50704	651714001	208284
8	668124530	500181562	557390911	5018	6474	5101	4753	503163999	47818
9	495615841	350862804	402479049	369	390	352	397	354972827	4110
10	342951117	222342651	268555704	4	16	7	4	226399116	
11	222591952	125300790	161615706					130227533	
12	130644906	58694151	85029441					62913393	
13	63457352	19257032	34491463					22365336	
14	19863504	3492819	8811640					4666095	
15	2528473	319918	1031310					495686	
16	137514	14858	56764					43766	
17	6396	576	3021						
18	439	1	225						
19	36		4						

Table C.56: Number of nonzeros per level for the weakly scaled 3D unstructured Laplacian on 256 processors.

C.5 3D Unstructured Anisotropic Problem

This section reports the results for experiments on a strongly scaled 7-point Laplacian problem. The problem is defined as

$$\begin{aligned} -(0.01u_{xx} + u_{yy} + 0.0001u_{zz}) &= 0 \quad \text{on } \Omega \qquad (\Omega = (0,1)^3), \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{C.2}$$

The sizes for this problem are identical to those in the 3D unstructured Laplacian in Section C.4. On one processor the problem has approximately approximately 211,000 unknowns. On 512 processors there is approximately 100 million unknowns, giving an average of 198,000 unknowns per processor. The individual trials and problem size growth for each trial are listed in Table C.57.

The data is organized into the following tables.

Trial information .... Table C.57

Grid complexities .... Table C.58

Relative grid complexities .... Table C.59

Operator complexities .... Table C.60

Relative operator complexities .... Table C.61

Amount of work per digit-of-accuracy .... Table C.62

Relative amount of work per digit-of-accuracy .... Table C.63

Convergence factors .... Table C.64

Setup times .... Table C.65

Relative setup times .... Table C.66

Level-by-level degrees of freedom .... Tables C.67 and C.68

Level-by-level nonzeros .... Tables C.69 and C.70

p	1	2	4	8	16	32	64	128	256	512
Relative n	1.00	2.56	4.95	7.66	18.41	37.12	60.26	139.09	285.77	478.92

Table C.57: Trials and relative trial sizes for the weakly scaled 3D unstructured anisotropic problem. The number of processors (p) is shown in the first row, and the number of unknowns (n) relative to the number of unknowns in the smallest trial is shown in the second row.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.86	2.06	2.06	2.00	2.05	2.02	1.90	1.94	1.93	1.80
CLJP	1.66	1.80	1.84	1.80	1.80	1.81	1.75	1.74	1.77	1.71
CLJP-c	1.66	1.81	1.84	1.80	1.81	1.82	1.76	1.75	1.77	1.70
PMIS	1.29	1.28	1.29	1.28	1.27	1.30	1.28	1.27	1.31	1.29
HMIS	1.30	1.29	1.31	1.30	1.29	1.32	1.31	1.28	1.34	1.32
PMIS-c1	1.29	1.28	1.29	1.28	1.27	1.30	1.29	1.27	1.32	1.30
PMIS-c2	1.29	1.28	1.29	1.28	1.27	1.30	1.28	1.26	1.31	1.29
CR-CLJP	1.66	1.80	1.84	1.80	1.80	1.81	1.75	1.74	1.77	1.71
CR-PMIS	1.29	1.28	1.23	1.28	1.27	1.24	1.28	1.26	1.24	1.29

Table C.58: Grid complexities for the weakly scaled 3D unstructured anisotropic problem.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.00	1.11	1.11	1.08	1.10	1.09	1.02	1.04	1.04	0.97
CLJP	1.00	1.09	1.11	1.08	1.09	1.09	1.06	1.05	1.07	1.03
CLJP-c	1.00	1.09	1.11	1.08	1.09	1.09	1.06	1.05	1.06	1.02
PMIS	1.00	0.99	1.00	0.99	0.98	1.01	1.00	0.98	1.02	1.00
HMIS	1.00	0.99	1.01	0.99	0.99	1.02	1.00	0.98	1.03	1.01
PMIS-c1	1.00	0.99	1.00	0.99	0.98	1.01	1.00	0.98	1.02	1.01
PMIS-c2	1.00	0.99	1.00	0.99	0.98	1.01	0.99	0.98	1.02	1.00
CR-CLJP	1.00	1.09	1.11	1.08	1.09	1.09	1.06	1.05	1.07	1.03
CR-PMIS	1.00	0.99	0.96	0.99	0.98	0.96	1.00	0.98	0.96	1.00

Table C.59: Grid complexities for the weakly scaled 3D unstructured anisotropic problem relative to the single processor grid complexities.

p	1	2	4	8	16	32	64	128	256	512
Falgout	5.34	7.64	7.55	6.82	8.25	8.35	6.75	8.18	8.73	6.47
CLJP	3.88	5.63	5.88	5.35	6.14	6.39	5.46	6.03	6.58	5.44
CLJP-c	3.97	5.87	6.20	5.62	6.47	6.80	5.75	6.45	7.10	5.71
PMIS	1.64	1.68	1.68	1.53	1.62	1.72	1.51	1.59	1.76	1.51
HMIS	1.70	1.76	1.78	1.61	1.72	1.85	1.59	1.69	1.92	1.61
PMIS-c1	1.65	1.68	1.69	1.53	1.62	1.73	1.51	1.60	1.79	1.53
PMIS-c2	1.64	1.68	1.68	1.53	1.62	1.72	1.50	1.59	1.75	1.49
CR-CLJP	3.83	5.63	5.92	5.37	6.14	6.41	5.46	6.03	6.58	5.44
CR-PMIS	1.64	1.67	1.46	1.53	1.61	1.48	1.51	1.59	1.50	1.51

Table C.60: Operator complexities for the weakly scaled 3D unstructured anisotropic problem.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.00	1.43	1.41	1.28	1.54	1.56	1.26	1.53	1.63	1.21
CLJP	1.00	1.45	1.52	1.38	1.58	1.65	1.41	1.55	1.70	1.40
CLJP-c	1.00	1.48	1.56	1.42	1.63	1.71	1.45	1.62	1.79	1.44
PMIS	1.00	1.02	1.02	0.93	0.98	1.05	0.92	0.97	1.07	0.92
HMIS	1.00	1.03	1.04	0.95	1.01	1.08	0.93	0.99	1.12	0.95
PMIS-c1	1.00	1.02	1.02	0.93	0.98	1.05	0.92	0.97	1.08	0.93
PMIS-c2	1.00	1.02	1.02	0.93	0.99	1.05	0.91	0.97	1.07	0.91
CR-CLJP	1.00	1.47	1.54	1.40	1.60	1.67	1.42	1.57	1.71	1.42
CR-PMIS	1.00	1.02	0.89	0.93	0.98	0.90	0.92	0.97	0.91	0.92

Table C.61: Operator complexities for the weakly scaled 3D unstructured anisotropic problem relative to the single processor operator complexities.

p	1	2	4	8	16	32	64	128	256	512
Falgout	97.59	168.11	162.04	186.59	291.42	280.15	269.25	365.18	467.90	383.69
CLJP	69.72	114.25	119.66	144.56	215.97	202.19	214.15	269.84	342.69	325.48
CLJP-c	73.73	124.92	127.75	151.74	225.46	218.11	227.33	288.52	374.08	335.50
PMIS	65.33	91.97	88.95	85.91	97.41	110.78	92.51	104.07	119.96	103.76
HMIS	66.33	89.28	93.14	82.08	102.94	116.04	97.61	107.79	127.01	110.66
PMIS-c1	67.96	91.30	92.42	79.90	97.85	110.53	92.49	104.08	121.28	105.08
PMIS-c2	66.19	89.59	88.42	80.85	97.90	110.57	91.44	103.25	121.31	102.99
CR-CLJP	67.97	115.50	120.27	144.42	215.70	202.53	214.86	270.08	342.88	325.38
CR-PMIS	65.69	92.40	159.16	81.51	96.67	154.32	92.13	100.77	156.19	113.77

Table C.62: Amount of work per digit-of-accuracy for the weakly scaled 3D unstructured anisotropic problem.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.00	1.72	1.66	1.91	2.99	2.87	2.76	3.74	4.79	3.93
CLJP	1.00	1.64	1.72	2.07	3.10	2.90	3.07	3.87	4.91	4.67
CLJP-c	1.00	1.69	1.73	2.06	3.06	2.96	3.08	3.91	5.07	4.55
PMIS	1.00	1.41	1.36	1.32	1.49	1.70	1.42	1.59	1.84	1.59
HMIS	1.00	1.35	1.40	1.24	1.55	1.75	1.47	1.63	1.91	1.67
PMIS-c1	1.00	1.34	1.36	1.18	1.44	1.63	1.36	1.53	1.78	1.55
PMIS-c2	1.00	1.35	1.34	1.22	1.48	1.67	1.38	1.56	1.83	1.56
CR-CLJP	1.00	1.70	1.77	2.12	3.17	2.98	3.16	3.97	5.04	4.79
CR-PMIS	1.00	1.41	2.42	1.24	1.47	2.35	1.40	1.53	2.38	1.73

Table C.63: Amount of work per digit-of-accuracy for the weakly scaled 3D unstructured anisotropic problem relative to single processor WPDA.

p	1	2	4	8	16	32	64	128	256	512
Falgout	0.78	0.81	0.81	0.85	0.88*	0.87*	0.89*	0.90*	0.92*	0.93*
CLJP	0.77	0.80	0.80	0.84	0.88*	0.86*	0.89*	0.90*	0.92*	0.93*
CLJP-c	0.78	0.81	0.80	0.84	0.88*	0.87*	0.89*	0.90*	0.92*	0.92*
PMIS	0.89*	0.92*	0.92*	0.92*	0.93*	0.93*	0.93*	0.93*	0.93*	0.94*
HMIS	0.89*	0.91*	0.92*	0.91*	0.93*	0.93*	0.93*	0.93*	0.93*	0.94*
PMIS-c1	0.89*	0.92*	0.92*	0.92*	0.93*	0.93*	0.93*	0.93*	0.93*	0.94*
PMIS-c2	0.89*	0.92*	0.92*	0.92*	0.93*	0.93*	0.93*	0.93*	0.94*	0.94*
CR-CLJP	0.77	0.80	0.80	0.84	0.88*	0.86*	0.89*	0.90*	0.92*	0.93*
CR-PMIS	0.89*	0.92*	0.96*	0.92*	0.93*	0.96*	0.93*	0.93*	0.96*	0.94*

Table C.64: Convergence factors for the weakly scaled 3D unstructured anisotropic problem. Asterisks (*) denote trials that did not converge to a relative residual smaller than 10^{-8} within 100 iterations.

p	1	2	4	8	16	32	64	128	256	512
Falgout	7.11	22.02	31.93	24.57	51.04	60.36	46.60	73.46	101.38	80.90
CLJP	5.65	17.66	25.69	19.91	38.61	46.02	37.01	53.79	71.53	63.73
CLJP-c	6.24	18.85	27.11	21.52	41.90	51.28	42.59	63.27	90.80	85.00
PMIS	2.48	5.65	6.64	4.78	7.93	9.06	6.47	8.40	10.71	8.45
HMIS	2.74	6.10	7.15	5.11	8.66	9.86	7.10	8.86	11.76	9.77
PMIS-c1	2.87	6.24	7.22	5.28	8.72	10.36	8.11	11.60	17.16	18.88
PMIS-c2	3.06	6.54	7.47	5.44	9.04	10.55	8.17	11.85	17.16	18.25
CR-CLJP	9.83	25.51	34.36	26.56	47.40	56.23	44.25	62.81	82.49	71.32
CR-PMIS	4.68	8.68	8.41	7.45	10.83	10.18	9.01	11.40	11.70	11.33

Table C.65: Setup times in seconds for the weakly scaled 3D unstructured anisotropic problem.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.00	3.09	4.49	3.45	7.17	8.48	6.55	10.33	14.25	11.37
CLJP	1.00	3.12	4.55	3.52	6.83	8.14	6.55	9.52	12.66	11.28
CLJP-c	1.00	3.02	4.34	3.45	6.72	8.22	6.82	10.14	14.55	13.62
PMIS	1.00	2.27	2.67	1.92	3.19	3.65	2.61	3.38	4.31	3.40
HMIS	1.00	2.23	2.61	1.87	3.17	3.60	2.59	3.24	4.30	3.57
PMIS-c1	1.00	2.18	2.52	1.84	3.04	3.61	2.83	4.05	5.99	6.59
PMIS-c2	1.00	2.14	2.44	1.78	2.95	3.45	2.67	3.87	5.61	5.96
CR-CLJP	1.00	2.59	3.49	2.70	4.82	5.72	4.50	6.39	8.39	7.25
CR-PMIS	1.00	1.85	1.80	1.59	2.31	2.18	1.92	2.44	2.50	2.42

Table C.66: Setup times for the weakly scaled 3D unstructured anisotropic problem relative to single processor setup times.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	211369	211369	211369	211369	211369	211369	211369	211369	211369
2	85102	72603	72834	49162	50837	49362	48984	72614	49153
3	44144	33944	34102	9792	10735	9868	9708	33942	9754
4	23904	16087	16276	2034	2346	2084	1970	16138	1974
5	13180	8215	8325	365	432	345	324	8274	321
6	7247	4273	4316	57	58	43	54	4323	
7	3873	2188	2211	7	9	7	6	2300	
8	1984	1106	1105					1213	
9	946	503	531						
10	401	217	237						
11	161	87	108						
12	57	31	38						
13	19	12	17						
14	7	6	7						

Table C.67: Number of degrees of freedom per level for the weakly scaled 3D unstructured anisotropic problem on one processor.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	101229153	101229153	101229153	101229153	101229153	101229153	101229153	101229153	101229153
2	35713055	33582858	33306669	22243356	23747507	22701817	21825775	33582997	22242940
3	19378970	17432708	17003107	5786193	6871307	6029836	5593538	17433800	5785603
4	11074486	9455842	9192350	1257156	1609354	1319186	1207623	9458445	1165532
5	6369163	5207559	5093003	218164	293028	227675	210436	5208667	203655
6	3651494	2883232	2856489	31672	44427	32827	30932	2884036	30007
7	2068960	1589311	1600100	3878	5707	3981	3878	1589313	5704
8	1146730	862347	887887	467	658	448	436	863523	846
9	621257	459340	486121	59	76	55	49	460030	83
10	330482	238992	261617	12	12	9	6	239233	
11	173689	120888	137751	2	3			121217	
12	90035	58991	70681					59318	
13	45428	27419	35080					27616	
14	22163	11953	16690					11999	
15	10340	4791	7452					4791	
16	4544	1696	3098					1702	
17	1788	555	1155						
18	676	162	412						
19	252	47	127						
20	95	20	39						
21	41	9	15						
22	17		5						
23	6								

Table C.68: Number of degrees of freedom per level for the weakly scaled 3D unstructured anisotropic problem on 512 processors.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	2763225	2763225	2763225	2763225	2763225	2763225	2763225	2763225	2763225
2	2212470	1736415	1783230	1270782	1346601	1283566	1267554	1736800	1270669
3	2241694	1705362	1755432	390424	451813	393474	386562	1709250	389050
4	2191236	1489583	1544136	101242	126390	104448	96970	1489926	99010
5	1945648	1210867	1261057	15707	20196	14355	12736	1199908	14473
6	1506171	882547	908242	1451	1344	899	1084	869719	
7	1014441	532166	549199	47	71	49	36	543234	
8	563840	266278	265189					283951	
9	235568	92519	101081						
10	72341	26837	30527						
11	17367	5979	9248						
12	2901	909	1360						
13	359	144	283						
14	49	36	49						

Table C.69: Number of nonzeros per level for the weakly scaled 3D unstructured anisotropic problem on one processor.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	1497203934	1497203934	1497203934	1497203934	1497203934	1497203934	1497203934	1497203934	1497203934
2	894915943	824379625	845526160	469610226	521004892	488327634	456579686	824386110	469603344
3	1040954307	916579833	944305184	205755198	270616290	218918282	198030590	916602103	205722135
4	1156156256	999403276	1031419673	69519409	98678842	73146012	67462775	999501750	64306507
5	1172105068	984749413	1023302695	14992931	22007460	15526179	14667568	984516843	14064515
6	1080515769	875017650	917497668	2156924	3218883	2182651	2156040	875110070	2057265
7	906362956	705952637	749549953	223258	341681	225331	228324	705411663	535444
8	692205094	517295101	561503094	19275	28332	18504	17104	518373853	61284
9	486283995	350339330	392794781	1271	1662	1145	981	351019082	2587
10	321474272	221883078	258023459	124	104	79	28	221557827	
11	201850509	130170084	159353927	4	9			130455593	
12	119078153	69180299	91096133					69786960	
13	63551398	32182621	47049124					32405940	
14	29979581	12265897	20980492					12243461	
15	11951100	3702873	7621052					3705007	
16	3969882	822058	2281512					853360	
17	1009002	146395	519789						
18	225800	17834	101268						
19	43506	2047	13387						
20	7489	400	1481						
21	1637	81	225						
22	287		25						
23	36								

Table C.70: Number of nonzeros per level for the weakly scaled 3D unstructured anisotropic problem on 512 processors.

C.6 3D Laplacian Holes

This section reports the results for experiments on a weakly scaled 3D unstructured Laplacian problem on the domain shown in Figure 4.21. The continuous problem is the same as Section C.4, and the problem is scaled to assign approximately the same number of unknowns to each processor for all trials. On one processor the problem receives approximately 380,000 unknowns. On 512 processors the problem has about 167 million unknowns, giving an average of 327,000 unknowns per processor. The individual trials and problem size growth for each trial are listed in Table C.71.

The data is organized into the following tables.

Trial information .... Table C.71

Grid complexities .... Table C.72

Relative grid complexities .... Table C.73

Operator complexities .... Table C.74

Relative operator complexities .... Table C.75

Amount of work per digit-of-accuracy .... Table C.76

Relative amount of work per digit-of-accuracy .... Table C.77

Convergence factors .... Table C.78

Setup times .... Table C.79

Relative setup times .... Table C.80

Level-by-level degrees of freedom .... Tables C.81 and C.82

Level-by-level nonzeros .... Tables C.83 and C.84

p	1	2	4	8	16	32	64	512
Relative n	1.00	3.06	4.98	7.25	19.14	37.15	55.84	439.75

Table C.71: Trials and relative trial sizes for the weakly scaled 3D unstructured Laplacian on the holes geometry. The number of processors (p) is shown in the first row, and the number of unknowns (n) relative to the number of unknowns in the smallest trial is shown in the second row.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.83	1.81	1.86	1.91	1.87	2.01	2.03	–	–	2.04
CLJP	1.65	1.63	1.68	1.72	1.67	1.79	1.83	–	–	1.91
CLJP-c	1.66	1.64	1.69	1.73	1.68	1.80	1.85	–	–	1.91
PMIS	1.21	1.21	1.21	1.22	1.21	1.24	1.25	–	–	1.26
HMIS	1.22	1.23	1.23	1.24	1.23	1.25	1.28	–	–	1.31
PMIS-c1	1.21	1.21	1.21	1.22	1.21	1.24	1.25	–	–	1.26
PMIS-c2	1.21	1.21	1.21	1.22	1.21	1.24	1.25	–	–	1.26
CR-CLJP	1.64	1.63	1.68	1.72	1.67	1.79	1.83	–	–	1.91
CR-PMIS	1.20	1.21	1.21	1.22	1.21	1.24	1.25	–	–	1.26

Table C.72: Grid complexities for the weakly scaled 3D unstructured Laplacian on the holes geometry.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.00	0.99	1.02	1.05	1.02	1.10	1.11	–	–	1.11
CLJP	1.00	0.99	1.02	1.04	1.01	1.09	1.11	–	–	1.16
CLJP-c	1.00	0.99	1.02	1.04	1.01	1.09	1.12	–	–	1.16
PMIS	1.00	1.00	1.01	1.01	1.00	1.02	1.03	–	–	1.04
HMIS	1.00	1.01	1.01	1.02	1.01	1.03	1.05	–	–	1.08
PMIS-c1	1.00	1.00	1.01	1.01	1.01	1.02	1.03	–	–	1.05
PMIS-c2	1.00	1.00	1.01	1.01	1.00	1.02	1.03	–	–	1.04
CR-CLJP	1.00	0.99	1.02	1.05	1.02	1.09	1.11	–	–	1.16
CR-PMIS	1.00	1.01	1.01	1.01	1.01	1.03	1.04	–	–	1.05

Table C.73: Grid complexities for the weakly scaled 3D unstructured Laplacian on the holes geometry relative to the single processor grid complexities.

p	1	2	4	8	16	32	64	128	256	512
Falgout	5.60	6.01	6.29	6.68	7.34	7.80	8.79	–	–	*
CLJP	4.33	4.68	4.96	5.16	5.61	6.44	6.96	–	–	*
CLJP-c	4.47	4.80	5.13	5.37	5.85	6.76	7.35	–	–	*
PMIS	1.43	1.47	1.45	1.45	1.52	1.48	1.50	–	–	1.52
HMIS	1.49	1.55	1.52	1.55	1.63	1.54	1.68	–	–	1.84
PMIS-c1	1.43	1.47	1.45	1.45	1.52	1.48	1.50	–	–	1.55
PMIS-c2	1.43	1.47	1.44	1.45	1.52	1.48	1.50	–	–	1.51
CR-CLJP	4.22	4.66	4.92	5.19	5.65	6.48	7.00	–	–	*
CR-PMIS	1.42	1.47	1.44	1.45	1.52	1.48	1.50	–	–	1.52

Table C.74: Operator complexities for the weakly scaled 3D unstructured Laplacian on the holes geometry. Operator complexities for Falgout, CLJP, CLJP-c, and CR-CLJP on 512 processors were corrupted due to overflow.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.00	1.07	1.12	1.19	1.31	1.39	1.57	–	–	*
CLJP	1.00	1.08	1.15	1.19	1.30	1.49	1.61	–	–	*
CLJP-c	1.00	1.07	1.15	1.20	1.31	1.51	1.64	–	–	*
PMIS	1.00	1.03	1.01	1.02	1.06	1.03	1.05	–	–	1.06
HMIS	1.00	1.04	1.03	1.05	1.10	1.04	1.13	–	–	1.24
PMIS-c1	1.00	1.03	1.01	1.02	1.06	1.04	1.05	–	–	1.08
PMIS-c2	1.00	1.03	1.01	1.02	1.06	1.03	1.05	–	–	1.05
CR-CLJP	1.00	1.10	1.17	1.23	1.34	1.54	1.66	–	–	*
CR-PMIS	1.00	1.03	1.02	1.02	1.07	1.04	1.05	–	–	1.07

Table C.75: Operator complexities for the weakly scaled 3D unstructured Laplacian on the holes geometry relative to the single processor operator complexities. Operator complexities for Falgout, CLJP, CLJP-c, and CR-CLJP on 512 processors were corrupted due to overflow.

p	1	2	4	8	16	32	64	128	256	512
Falgout	11.60	15.09	15.85	17.59	21.72	24.91	30.00	–	–	*
CLJP	9.02	11.82	12.77	13.89	16.83	21.13	23.96	–	–	*
CLJP-c	9.36	12.18	13.22	14.54	17.68	22.33	25.45	–	–	*
PMIS	8.71	14.33	14.86	17.75	26.19	30.17	28.37	–	–	41.83
HMIS	8.73	13.00	13.11	15.91	20.41	23.20	27.77	–	–	46.79
PMIS-c1	8.64	15.06	15.62	17.47	26.16	25.32	28.69	–	–	42.95
PMIS-c2	8.71	14.61	14.87	17.92	23.04	25.92	28.40	–	–	41.31
CR-CLJP	14.81	25.42	38.07	32.09	35.70	45.54	41.80	–	–	*
CR-PMIS	12.24	18.80	23.08	28.78	28.81	37.44	45.21	–	–	44.73

Table C.76: Amount of work per digit-of-accuracy for the weakly scaled 3D unstructured Laplacian on the holes geometry. Overflow on 512 processors for the Falgout, CLJP, CLJP-c, and CR-CLJP tests corrupted the WPDA results.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.00	1.30	1.37	1.52	1.87	2.15	2.59	–	–	*
CLJP	1.00	1.31	1.42	1.54	1.87	2.34	2.66	–	–	*
CLJP-c	1.00	1.30	1.41	1.55	1.89	2.39	2.72	–	–	*
PMIS	1.00	1.65	1.71	2.04	3.01	3.46	3.26	–	–	4.80
HMIS	1.00	1.49	1.50	1.82	2.34	2.66	3.18	–	–	5.36
PMIS-c1	1.00	1.74	1.81	2.02	3.03	2.93	3.32	–	–	4.97
PMIS-c2	1.00	1.68	1.71	2.06	2.64	2.98	3.26	–	–	4.74
CR-CLJP	1.00	1.72	2.57	2.17	2.41	3.08	2.82	–	–	*
CR-PMIS	1.00	1.54	1.89	2.35	2.35	3.06	3.69	–	–	3.65

Table C.77: Amount of work per digit-of-accuracy for the weakly scaled 3D unstructured Laplacian on the holes geometry relative to single processor WPDA. Overflow on 512 processors for the Falgout, CLJP, CLJP-c, and CR-CLJP tests corrupted the WPDA results.

p	1	2	4	8	16	32	64	128	256	512
Falgout	0.11	0.16	0.16	0.17	0.21	0.24	0.26	–	–	0.33
CLJP	0.11	0.16	0.17	0.18	0.22	0.25	0.26	–	–	0.32
CLJP-c	0.11	0.16	0.17	0.18	0.22	0.25	0.26	–	–	0.32
PMIS	0.47	0.62	0.64	0.69	0.77	0.80	0.78	–	–	0.85*
HMIS	0.46	0.58	0.59	0.64	0.69	0.74	0.76	–	–	0.83*
PMIS-c1	0.47	0.64	0.65	0.68	0.77	0.76	0.79	–	–	0.85*
PMIS-c2	0.47	0.63	0.64	0.69	0.74	0.77	0.78	–	–	0.85*
CR-CLJP	0.28	0.43	0.56	0.48	0.48	0.52	0.46	–	–	0.40
CR-PMIS	0.59	0.70	0.75	0.79	0.78	0.83*	0.86*	–	–	0.86*

Table C.78: Convergence factors for the weakly scaled 3D unstructured Laplacian on the holes geometry. Asterisks (*) denote trials that did not converge to a relative residual smaller than 10^{-8} within 100 iterations.

p	1	2	4	8	16	32	64	128	256	512
Falgout	16.55	48.04	41.61	34.80	60.41	73.79	67.71	–	–	134.37
CLJP	15.03	43.67	36.68	29.24	48.42	63.66	56.67	–	–	108.49
CLJP-c	16.24	46.08	39.00	31.39	52.45	68.77	62.50	–	–	140.14
PMIS	4.81	13.80	10.23	7.44	14.07	13.12	10.68	–	–	13.05
HMIS	5.44	15.65	11.47	8.46	14.07	14.13	12.50	–	–	16.76
PMIS-c1	5.59	15.36	11.41	8.41	14.27	15.09	13.25	–	–	29.55
PMIS-c2	6.04	16.47	12.07	8.87	14.99	15.86	13.80	–	–	29.47
CR-CLJP	21.92	60.40	49.60	40.19	65.57	80.75	71.45	–	–	136.87
CR-PMIS	8.38	22.14	15.80	11.68	19.12	18.78	15.04	–	–	19.18

Table C.79: Setup times in seconds for the weakly scaled 3D unstructured Laplacian on the holes geometry.

p	1	2	4	8	16	32	64	128	256	512
Falgout	1.00	2.90	2.51	2.10	3.65	4.46	4.09	–	–	8.12
CLJP	1.00	2.90	2.44	1.95	3.22	4.23	3.77	–	–	7.22
CLJP-c	1.00	2.84	2.40	1.93	3.23	4.23	3.85	–	–	8.63
PMIS	1.00	2.87	2.13	1.55	2.93	2.73	2.22	–	–	2.71
HMIS	1.00	2.88	2.11	1.56	2.59	2.60	2.30	–	–	3.08
PMIS-c1	1.00	2.75	2.04	1.51	2.55	2.70	2.37	–	–	5.29
PMIS-c2	1.00	2.73	2.00	1.47	2.48	2.63	2.29	–	–	4.88
CR-CLJP	1.00	2.76	2.26	1.83	2.99	3.68	3.26	–	–	6.24
CR-PMIS	1.00	2.64	1.89	1.39	2.28	2.24	1.79	–	–	2.29

Table C.80: Setup times for the weakly scaled 3D unstructured Laplacian on the holes geometry relative to single processor setup times.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	380822	380822	380822	380822	380822	380822	380822	380822	380822
2	154583	134930	135206	64117	66798	64051	63963	134958	64107
3	81765	61070	61663	12331	13651	12316	12303	61885	12005
4	42114	28126	28628	2236	2486	2219	2247	29043	1464
5	20892	12997	13410	344	423	359	359	13529	
6	9871	5825	6176	66	77	61	60	5301	
7	4422	2479	2732	17	23	15	18		
8	1793	1004	1102	6	6	4	5		
9	676	358	394						
10	241	133	143						
11	98	56	58						
12	45	28	24						
13	19	11	10						
14	9	6	5						

Table C.81: Number of degrees of freedom per level for the weakly scaled 3D unstructured Laplacian on the holes geometry on one processor.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	21264640	21264640	21264640	21264640	21264640	21264640	21264640	21264640	21264640
2	10017626	8963775	9067947	4348272	4881453	4325417	4339394	8963939	4348011
3	5537706	4349270	4443056	750491	873417	751400	747151	4349509	750416
4	3054822	2179732	2249119	120048	142912	120560	119634	2180831	119384
5	1635178	1100065	1152498	18143	22184	18190	18052	1101803	18025
6	845102	554950	591817	2569	3276	2556	2548	557358	2226
7	421903	275498	299617	376	472	362	351	278804	
8	202886	132032	146767	60	86	57	52	136381	
9	93708	60311	68605	18	24	16	15	66255	
10	41231	25761	30114	6	7	6	6	29411	
11	17046	10136	12204					11110	
12	6561	3523	4496						
13	2279	1095	1463						
14	734	326	440						
15	244	125	146						
16	91	54	62						
17	41	21	25						
18	22	11	11						
19	12	5	5						
20	9								

Table C.82: Number of degrees of freedom per level for the weakly scaled 3D unstructured Laplacian on the holes geometry on 64 processors.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	4143920	4143920	4143920	4143920	4143920	4143920	4143920	4143920	4143920
2	3982565	3582378	3649860	1330639	1456532	1334767	1329763	3582700	1330511
3	4726471	3828588	3931767	374837	462825	376170	374387	3865289	367579
4	4351556	2969692	3076856	65652	80878	65309	66451	3042517	41628
5	3079558	1903555	2013798	7018	9809	7189	7255	2000799	
6	1756545	980563	1085578	1002	1143	779	754	840485	
7	808842	386725	456644	143	275	107	178		
8	279917	122348	141514	32	34	16	19		
9	73976	28730	33310						
10	15851	6497	7319						
11	4244	1584	1860						
12	1257	520	440						
13	321	113	88						
14	81	36	25						

Table C.83: Number of nonzeros per level for the weakly scaled 3D unstructured Laplacian on the holes geometry on one processor.

Level	Falgout	CLJP	CLJP-c	PMIS	HMIS	PMIS-c1	PMIS-c2	CR-CLJP	CR-PMIS
1	299051224	299051224	299051224	299051224	299051224	299051224	299051224	299051224	299051224
2	347523970	320915673	326141975	112115860	153445975	111360803	111878604	320919011	112111665
3	418620736	349869222	360329420	29987433	41936811	30146824	29801311	349802199	29995862
4	430244226	327735220	344226693	5735236	7935576	5795606	5693272	327757043	5709668
5	378666184	271397737	290638812	818435	1141966	828932	811404	271448381	816069
6	294093430	209552344	228719083	90303	130242	89912	89410	209887508	79278
7	207598535	146139470	163228283	8966	12418	8238	8165	147446508	
8	131612812	87792102	101011779	834	1420	811	734	90357449	
9	71814270	44190075	52703503	214	262	134	117	48999873	
10	32677083	17761507	22273780	34	45	32	36	21661639	
11	11970218	5714036	7517970					6338766	
12	3552465	1346481	1968976						
13	800063	239059	392385						
14	143934	35558	62574						
15	24002	7443	10774						
16	5009	1832	2464						
17	1335	379	531						
18	444	121	119						
19	142	25	25						
20	79								

Table C.84: Number of nonzeros per level for the weakly scaled 3D unstructured Laplacian on the holes geometry on 64 processors.

References

[1] D. Alber. Computational local Fourier mode analysis in the multigrid solution of coupled systems. Master's thesis, University of Illinois at Urbana-Champaign, 2004.

[2] D. Alber. Modifying CLJP to select grid hierarchies with lower operator complexities and better performance. *Numerical Linear Algebra with Applications*, 13:87–104, 2006.

[3] D. Alber and L. Olson. Parallel coarse grid selection. *Numerical Linear Algebra with Applications*, accepted, 2007.

[4] N. S. Bakhvalov. On the convergence of a relaxation method with natural constraints on the elliptic operator. *USSR Comp. Math. Math. Phys.*, 6:101–135, 1966.

[5] M. Benzi and M. Tůma. A comparative study of sparse approximate inverse preconditioners. In *IMACS'97: Proceedings on the on Iterative methods and preconditioners*, pages 305–340, New York, NY, USA, 1999. Elsevier North-Holland, Inc.

[6] B. Bollobás. *Graph Theory*. Springer-Verlag, New York, 1979.

[7] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Math. Comput.*, 31(138):333–390, 1977.

[8] A. Brandt. General highly accurate algebraic coarsening. *Electronic Transactions on Numerical Analysis*, 10:1–20, 2000.

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

[10] J. Brannick. *Adaptive Algebraic Multigrid Coarsening Strategies*. PhD thesis, University of Colorado at Boulder, 2005.

[11] 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.

[12] D. Brélaz. New methods to color the vertices of a graph. *Commun. ACM*, 22(4):251–256, 1979.

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

[14] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial*. Society for Industrial and Applied Mathematics, Philadelphia, second edition, 2000.

[15] J. S. Butler. Improving coarsening and interpolation for algebraic multigrid. Master's thesis, University of Waterloo, 2006.

[16] U. V. Catalyurek, E. G. Boman, K. D. Devine, D. Bozdag, R. T. Heaphy, and L. A. Riesen. Hypergraph-based dynamic load balancing for adaptive scientific computations. In *Proc. of 21st International Parallel and Distributed Processing Symposium (IPDPS'07)*. IEEE, 2007.

[17] E. Chow, R. D. Falgout, J. J. Hu, R. S. Tuminaro, and U. M. Yang. A survey of parallelization techniques for multigrid solvers. Technical report, Lawrence Livermore National Laboratory, August 2004. UCRL-BOOK-205864.

[18] A. J. Cleary, R. D. Falgout, V. E. Henson, and J. E. Jones. Coarse-grid selection for parallel algebraic multigrid. In *Proceedings of the 5th International Symposium on Solving Irregularly Structured Problems in Parallel*, pages 104–115. Springer-Verlag, 1998.

[19] A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, G. N. Miranda, and J. W. Ruge. Robustness and scalability of algebraic multigrid. *SIAM J. Sci. Comput.*, 21(5):1886–1908, 2000.

[20] G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. *J. Parallel Distrib. Comput.*, 7(2):279–301, 1989.

[21] H. L. deCougny, K. D. Devine, J. E. Flaherty, R. M. Loy, C. Özturan, and M. S. Shephard. Load balancing for the parallel adaptive solution of partial differential equations. *Appl. Numer. Math.*, 16(1-2):157–182, 1994.

[22] K. D. Devine, E. G. Boman, R. T. Heaphy, R. H. Bisseling, and U. V. Catalyurek. Parallel hypergraph partitioning for scientific computing. In *Proc. of 20th International Parallel and Distributed Processing Symposium (IPDPS'06)*. IEEE, 2006.

[23] K. D. Devine, E. G. Boman, and G. Karypis. Partitioning and load balancing for emerging parallel applications and architectures. In M. Heroux, A. Raghavan, and H. Simon, editors, *Frontiers of Scientific Computing*. SIAM, Philadelphia, 2006.

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

[25] P. Vaněk, J. Mandel, and M. Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56(3):179–196, 1996.

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

[27] R. D. Falgout, P. S. Vassilevski, and L. T. Zikatanov. On two-grid convergence estimates. *Numerical Linear Algebra with Applications*, 12(5–6):471–494, 2005.

[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, Part III*, volume 2331 of *Lecture Notes in Computer Science*, pages 632–641. Springer-Verlag, 2002.

[29] R. D. Falgout and U. M. Yang. hypre: A library of high performance preconditioners. In *ICCS '02: Proceedings of the International Conference on Computational Science-Part III*, pages 632–641, London, UK, 2002. Springer-Verlag.

[30] R. P. Fedorenko. A relaxation method for solving elliptic difference equations. *USSR Comp. Math. Math. Phys.*, 1(5):1092–1096, 1962.

[31] Y. Fu. Dominating set and converse dominating set of a directed graph. *The American Mathematical Monthly*, 75(8):861–863, Oct. 1968.

[32] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W. H. Freeman, New York, 1979.

[33] A. George. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, 10(2):345–363, Apr 1973.

[34] A. George and W. H. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Rev.*, 31(1):1–19, 1989.

[35] P. F. Gorder. Multicore processors for science and engineering. *Comput. Sci. Eng.*, 9(2):3–7, 2007.

[36] M. Griebel, B. Metsch, D. Oeltz, and M. A. Schweitzer. Coarse grid classification: a parallel coarsening scheme for algebraic multigrid methods. *Numerical Linear Algebra with Applications*, 13(2–3):193–214, 2006.

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

[38] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of National Bureau Standards*, 49(6):409–436, December 1952.

[39] M. T. Jones and P. E. Plassmann. A parallel graph coloring heuristic. *SIAM Journal on Scientific Computing*, 14(3):654–669, 1993.

[40] M. T. Jones and P. E. Plassmann. Scalable iterative solution of sparse linear systems. *Parallel Computing*, 20(5):753–773, 1994.

[41] G. Karypis, K. Schloegel, and V. Kumar. *ParMETIS: Parallel Graph Partitioning and Sparse Matrix Ordering Library*, 2003.

[42] T. Kolev. aFEM software library, 2006.

[43] A. Krechel and K. Stüben. Parallel algebraic multigrid based on subdomain blocking. *Parallel Computing*, 27:1009–1031, 2001.

[44] O. E. Livne. Coarsening by compatible relaxation. *Numerical Linear Algebra with Applications*, 11:205–227, 2004.

[45] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Compututing*, 15(4):1036–1055, 1986.

[46] M. T. Heath, E. Ng, and B. W. Peyton. Parallel algorithms for sparse linear systems. *SIAM Review*, 33(3):420–460, Sep 1991.

[47] S. MacLachlan and Y. Saad. A greedy strategy for coarse-grid selection. *SIAM J. Sci. Comp.*, to appear, 2006.

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

[49] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003.

[50] Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, 1986.

[51] K. Schloegel, G. Karypis, and V. Kumar. A unified algorithm for load-balancing adaptive scientific simulations. In *Supercomputing '00: Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM)*, page 59, Washington, DC, USA, 2000. IEEE Computer Society.

[52] H. De Sterck, U. M. Yang, and J. J. Heys. Reducing complexity in parallel algebraic multigrid preconditioners. *SIAM Journal on Matrix Analysis and Applications*, 27:1019–1039, 2006.

[53] K. Stüben. An introduction to algebraic multigrid. In *Multigrid*, pages 413–532. Academic Press, 2000.

[54] T. F. Coleman and J. J. More. Estimation of sparse Jacobian matrices and graph coloring problems. *SIAM Journal on Numerical Analysis*, 20(1):187–209, feb 1983.

[55] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, London, 2001.

[56] W. L. Wan, T. F. Chan, and B. Smith. An energy-minimizing interpolation for robust multigrid methods. *SIAM J. Sci. Comput.*, 21(4):1632–1649, 2000.

[57] P. Wesseling. *An Introduction to Multigrid Methods*. R.T. Edwards, Flourtown, Pennsylvania, 2004.

[58] Roman Wienands. *Extended Local Fourier Analysis for Multigrid: Optimal Smoothing, Coarse Grid Correction, and Preconditioning*. PhD thesis, Universität zu Köln, 2001.

[59] Roman Wienands. *Manual for the Local Fourier Analysis Program LFA_2D_scalar: General Scalar Equations in Two Dimensions*. FhG - Institute for Algorithms and Scientific Computing (SCAI), D-53754 Sankt Augustin, Germany, October 2001.

[60] Roman Wienands and Cornelis W. Oosterlee. On three-grid Fourier analysis for multigrid. *SIAM J. Sci. Comput.*, 23(2):651–671, July 2001.

[61] Roman Wienands, Cornelis W. Oosterlee, and Takumi Washio. Fourier analysis of GMRES(m) preconditioned by multigrid. *SIAM J. Sci. Comput.*, 22(2):582–603, August 2000.

[62] J. Xu and L. Zikatanov. On an energy minimizing basis for algebraic multigrid methods. *Computing and Visualization in Science*, 7(3–4):121–127, October 2004.

Author's Biography

David Michael Alber was born in Iowa City, Iowa on June 14, 1977. He earned a Bachelor of Science degree in Biology and Computer Science from the University of Iowa in 1999 with distinction and honors in Computer Science. Following his undergraduate education, he joined the Department of Computer Science at the University of Illinois at Urbana-Champaign. Alber earned a Master of Science in Computer Science in 2004.