A GPU-BASED MAXIMAL INDEPENDENT SET AGGREGATION STRATEGY: ALGORITHMS, COMPARISONS, AND APPLICATIONS WITHIN ALGEBRAIC MULTIGRID

by

Thomas James Lewis

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

in

Computing

School of Computing

The University of Utah

August 2014

The University of Utah Graduate School

STATEMENT OF THESIS APPROVAL

The thesis of **Thomas James Lewis**

has been approved by the following supervisory committee members:

Robert M. Kirby	, Chair	**5/29/2014** Date Approved
Ross T. Whitaker	, Member	**5/29/2014** Date Approved
Mary Hall	, Member	**5/30/2014** Date Approved

and by **Ross T. Whitaker**, Chair/Dean of

the Department/College/School of **Computing**

and by David B. Kieda, Dean of The Graduate School.

ABSTRACT

The algebraic multigrid (AMG) method is often used as a preconditioner in Krylov subspace solvers such as the conjugate gradient method. An AMG preconditioner hierarchically aggregates the degrees of freedom during the coarsening phase in order to efficiently account for lower-frequency errors. Each degree of freedom in the coarser level corresponds to one of the aggregates in the finer level. The aggregation in each level in the hierarchy has a significant impact on the effectiveness of AMG as a preconditioner. The aggregation can be formulated as a partitioning problem on the graph induced from the matrix representation of a linear system.

The contributions of this work are as follows: first, a GPU implementation of a "bottom-up" partitioning scheme based on maximal independent sets (MIS), including an efficient conditioning scheme for enforcing partition size constraints; second, three novel topological metrics, convexity, eccentricity, and minimum enclosing ball, for measuring partition quality; third, empirical test results comparing our MIS-Based aggregation methods with the MeTis graph partioning library, showing that the metrics correlate more strongly with AMG performance than the commonly used edge-cut metric, and that for finer aggregations, MIS-based aggregation is better suited for AMG coarsening than is the "top down" MeTis graph partitioning library, but that for coarser aggregations, MeTis performs better.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

This thesis is the culmination of a larger body of work which began with my initial work implementing aggregation for algebraic multigrid (AMG) coarsening, on Nvidia GPUs, as a component of the finite element method (FEM) pipeline detailed in Fu et al. [4]. We observed that our AMG implementation performed better using aggregators based on k-maximal independent sets (MIS(k)), compared with using the MeTis graph partitioning library [6] for aggregation. Our serial implementation of the FEM solver using an aggregator based on the flood-fill MIS(k) algorithm converged to a solution in fewer iterations than the GPU implementation using an aggregator based on the randomized MIS(k) algorithm. In addition, while the serial aggregator produced aggregations of sufficient regularity to meet the size constraints of the GPU-based AMG solver, the GPU-based aggregator did not. This required implementing conditioning techniques to enforce size constraints on the aggregations.

This motivated further investigation into MIS(k) algorithms, and the properties of aggregations produced using them, detailed in my bachelor's thesis [8]. Through that work, it became apparent that a commonly used measure of graph-partitioning performance, edge-cut, did not seem to capture the observed difference in AMG performance; the metrics of average internal valence and internal to external nodes ratio introduced also failed to measure the observed qualitative differences effectively. Using the node exchange conditioner developed for the FEM solver to enforce tighter constraints on aggregate size revealed that it did not always maintain the validity of the aggregation being conditioned, the modifications required to guarantee the aggregation remained valid made the conditioner unnacceptably slow.

In this work, I build on the foundations of my previous work and introduce three novel metrics of aggregation quality, based on adapting the geometric concepts of convexity and eccentricity for application to graph topology, a more advanced

aggregation conditioner, based on merging and splitting aggregates. This thesis is an expanded version of a paper of the same title submitted to the IPDPS-PDSEC conference workshop. Dr. Shankar Sastry was a co-author of that paper, contributing his writing talents to portions of the introduction, background, and results portions of the paper. With his permission I have modified and expanded these parts in my thesis. Because this work is so closely related with my bachelor's thesis, there is considerable overlap in parts of the background and related work sections, for the sake of completeness, I have included the overlap in this document, portions of which are duplicated in my bachelor's thesis.

My work detailed in my Bachelors Thesis and in Fu et al., much of which forms the foundation for the contributions in this document, was supported by an NSF REU grant under NSFIIS-0914564, while I was an undergraduate student. My continuation of the work as a graduate student was supported by the DOE NET DE-EE0004449 grant and DOE NET DE-EE0004449 grant. I would like to thank Dr. Shankar Sastry for his assistance writing the paper on which this thesis is based and for his mentorship, guidance, and commentary throughout the process of producing this document. Dr. Zhisong Fu wrote most of the FEM solver used in the testing, provided the meshes used in our experiments, and was an excellent collaborator while I was an inexperienced undergraduate researcher. My advisor, Dr. Mike Kirby, has given me excellent advice, assistance, mentorship, and above all, patience through both my undergraduate and graduate research careers and has helped me immeasurably in developing as a competent researcher; he will always have my sincerest gratitude. Lastly, I would like to thank my wife, Aubrey, for her unfailing faith and support, even at the times I doubted myself.

CHAPTER 1

INTRODUCTION

In numerical simulation of physical systems, there is a constant demand for increased computational efficiency, as any increase in computational efficiency allows the simulation of systems in greater detail, or the simulation of more complicated systems. The finite element method (FEM) is a widely used numerical method for approximating the solution of partial differential equations over irregular domains. A significant portion of the work in the finite element method is finding the solution of a large, sparse system of linear equations.

Numerical solutions of large, sparse linear systems are commonly obtained using iterative relaxation algorithms such as the Jacobi method. Such techniques typically are highly effective at reducing high-frequency errors in the solution, but for removing lower frequency errors, these techniques are rather inefficient, motivating the development of hierarchical methods such as multigrid.

Algebraic multigrid (AMG) is an iterative, hierarchical method for solving systems of linear equations. In AMG, a hierarchy of smaller linear systems is constructed, each of which approximates the original system, along with prolongation and restriction operators to transfer vectors from one level to another. AMG works because smooth, low-frequency, errors in the original system become higher frequency errors in the coarser systems, or grids, and can be effectively removed in the coarse level by simple relaxation operators.

One variant of AMG, smoothed aggregation multigrid (SAMG), constructs the hierarchy of linear systems by aggregating the degrees of freedom of the fine level linear system together into connected disjoint aggregates. The coarser level system is then composed with a degree of freedom corresponding to each aggregate. In smoothed aggregation multigrid, the aggregation chosen uniquely defines the linear

system, prolongator, and restrictor, for each level of the hierarchy. This implies that the number of iterations required to reach a solution depends only on the aggregation given, so finding a "good" aggregation is critical. It has been observed that AMG methods are very efficient when used as a preconditioner for the linear conjugate gradient solver (PC-AMG) [12] and other Krylov subspace methods. The AMG method is explained in more detail in Chapter 2.

A maximal independent set (MIS) is a set of pairwise disconnected nodes, in a graph which is maximal, meaning no other nodes may be added to it without breaking the independence property. An MIS(k) generalizes an MIS to have a radius of independence of k; that is, two nodes are considered connected if there exists a path between them of length k or less. A more detailed treatment of MIS and MIS(k) is included in Chapter 2.

The motivation for this work, as well as that in Lewis [8], came about from the investigation of SAMG, as part of an FEM pipeline, as detailed in Fu et al.[4]. The problem of finding an aggregation was treated as a graph partitioning problem, in which the degrees of freedom in the linear system are nodes in the graph, and the edges between nodes correspond to non-zero entries in the matrix. Using the general graph partitioning library MeTis [6] for aggregation, we observed that when specifying a desired number of partitions large enough to produce an optimal coarsening for multigrid, the convergence rate did not improve as predicted by the mathematical theory. This led us to implement a partitioning scheme based on maximal independent sets (MIS), similar to that described in Tuminaro et al. [13]. This strategy significantly improved the number of AMG cycles required, leading us to conclude that the aggregation produced was better suited for SAMG than the partition produced with MeTis.

Visually comparing aggregations produced by our MIS method with those produced using MeTis shows noticeable qualitative differences, which become more pronounced at each level in the hierarchy. Figure 1.1 shows hierarchies of aggregations of an irregular triangular mesh, for both an MIS-based aggregator (Figure 1.1 a, b, c) and MeTis (Figure 1.1 d, e, f). Nodes of the mesh are drawn as colored balls, the color

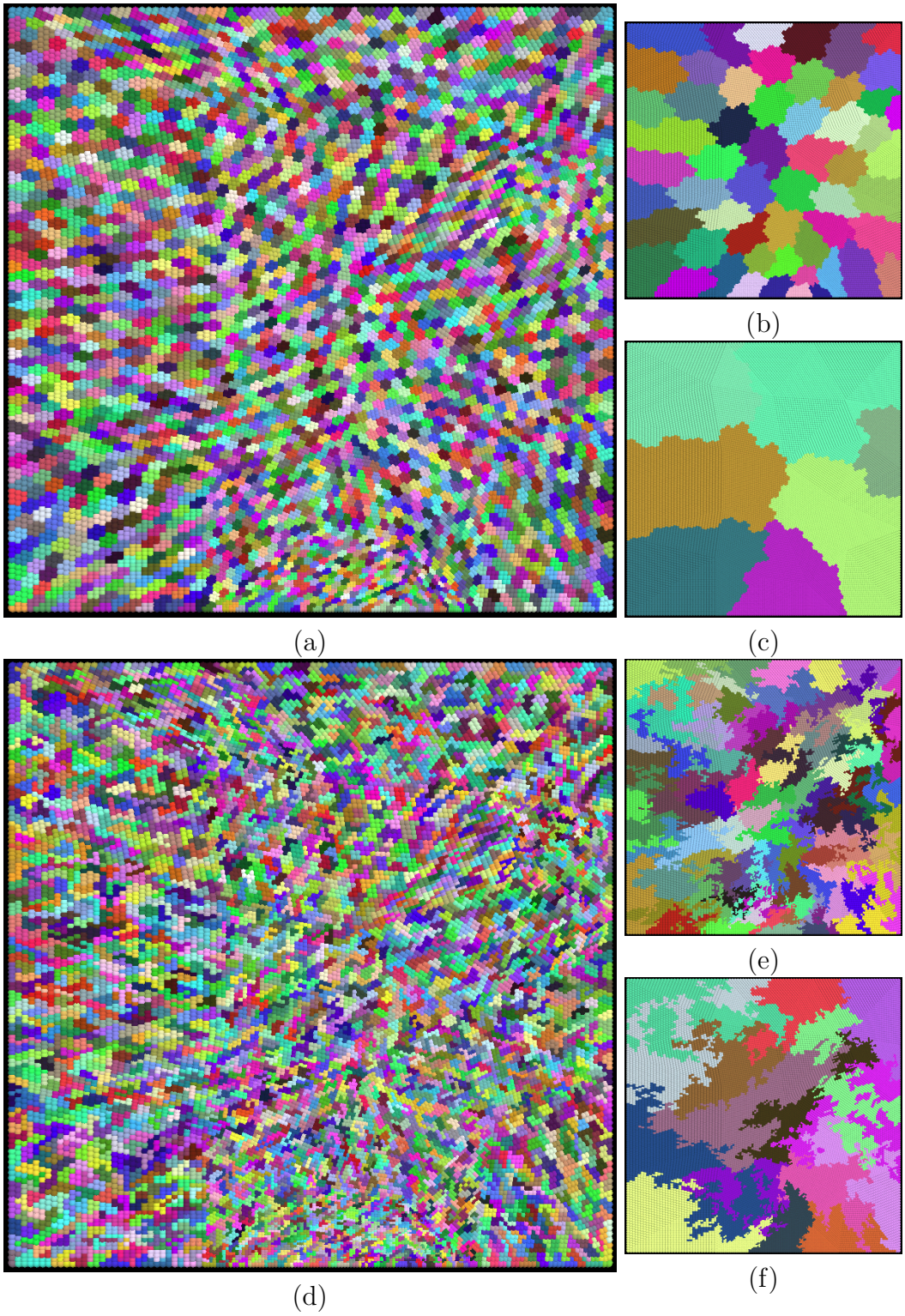


Figure 1.1. MIS and MeTis aggregation hierarchies: MIS(2) level 1,(a), level 3,(b), and level 4,(c). MeTis level 1,(d), level 3,(e), level 4(f).

indicating the aggregate they belong to. For higher levels in the hierarchy, the coloring of the induced mesh is projected back to the original mesh nodes. The first (Figure 1.1 a, d), third (Figure 1.1 b, e), and fourth levels (Figure 1.1 c, f) of the hierarchies are shown. Qualitatively, the aggregates produced with the MIS method appear rounder and more compact compared to those in the MeTis-based aggregation, with the boundaries between aggregates smoother. The MIS-based aggregator maintains the smooth shape of the aggregates through each successive aggregation, while the MeTis aggregator does not.

While the MIS-based aggregations are composed of more regularly shaped aggregates, they differ more in size than those produced with MeTis. This can cause issues in applications where there are strict size constraints. In Fu et al. [4], the node exchange conditioning method was introduced for adjusting the aggregate sizes in an aggregation, and examined in more detail in Lewis [8], where it was found to be lacking in robustness and general applicability.

This thesis addresses the following questions:

a) How does one evaluate the quality of an aggregate, i.e., are there heuristic metrics to determine if an aggregate positively affects the performance of the PCG-AMG method?

b) How are the metrics correlated with the solution time and the number of iterations?

c) How can an aggregation which does not meet aggregate size constraints be conditioned to do so efficiently?

To answer these questions, we employ two kinds of techniques to aggregate the graph nodes: a) Maximal independent set (MIS)-based and b) MeTis-based method. The aggregation methods are described in Chapter 3. We use the FEM and PCG-AMG solver with each aggregator to solve the elliptic Helmholtz equation, on four meshes, and compare the solution times, PCG-AMG iterations, and metric scores.

The testing methodology and description of our aggregation quality metrics are provided in Chapter 4. In practical applications, note that the linear solver may be

used multiple times on the same mesh. Consequently, improvements in solution times have a very significant impact on the total time required for a real-world simulation. The results from our numerical experiments show that our quality metrics do correlate with the solution time and the total number of iterations. These results are provided in Chapter 5. Conclusions and future research directions are given in Chapter 6.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, we provide some background information on the MeTis graph partioning library, maximal independent sets, the finite element method, and multigrid methods.

2.1 MeTis

MeTis [6] is a graph partitioning library that uses a multilevel algorithm involving three distinct phases: a) coarsening, b) initial partitioning, and c) uncoarsening. In the coarsening phase, the graph nodes are aggregated together hierarchically until the coarsened graph is small. Then, initial partitions are computed using very efficient methods such as spectral partitioning [5], matching, or clustering. These partitions are then projected back to finer graphs while simultaneously being optimized for the number of edges that cross from one partition to another. The optimization is carried out using the Kernighan-Lin method [7], the Fiduccia-Mattheyses method [3], or other methods. We consider this a "top-down" approach because the partitioning phase occurs at the coarsest level and is then projected back on to the finer levels. MeTis is designed to optimize the number of edges crossing the partitions. For our purpose, we use *kMeTis*, which uses a recursive bipartitioning technique to compute as many partitions as specified by the user.

For our application, we observe that MeTis does not perform well when used to create aggregations composed of very small groupings because is it based on the recursive bipartitioning technique. As many recursive calls are needed to produce the large number of small aggregates, and each call introduces more variability in partition size, we obtained aggregates with large variations in their sizes. In some cases, the aggregates were disconnected.

2.2 Maximal Independent Sets

We now proceed to a description of maximal independent sets. For purposes of illustration, we introduce a simple graph (Figure 2.1), which is the nodal graph of a simple mesh that might be obtained by the triangularization of an irregular point set on a planar domain. This is an example of the type of mesh commonplace in FEM. While we choose a 2D example as a visualization tool, all the concepts presented extend to 3D tetrahedral meshes, as well as more general graphs.

A maximal independent set (MIS) is a set of nodes x in a graph $G(v, e)$ that are independent, that is, no pair of nodes in the set is connected by an edge. More formally, for all (a, b) where $a \in x$ and $b \in x, (a, b) \notin e$. The set is also maximal, meaning that for every node $n \in v$, $n \notin x$ adding n to the set x would break the independence property. Examples of an MIS in our simple graph are shown in Figure 2.2.

In this work, as in Bell et al. [1], we generalize the idea of MIS to MIS(k) where the parameter k specifies the required radius of independence. This changes the independence condition from there being no pair of nodes in the set connected by an edge to no pair of nodes in the set that has a shortest path between them of length k. For a set x of nodes to be a valid MIS(k) of a graph $G(v, e)$, it must be true that for every (a, b) where $a \in x$ and $b \in x$ the shortest path between a and b is greater than k. The condition for maximality remains the same. Examples of a 2-MIS in our simple graph are shown in Figure 2.3.[1]

2.3 Finite Element Method

The finite element method (FEM) is a technique for numerically approximating solutions of partial differential equations (PDEs). It restricts the problem to a finite dimensional function space and finds the best approximation of the unknown function within that space. A significant advantage of FEM is that it works on unstructured meshes discritizing the spatial domain, rather than a structured grid or lattice, which

[1]The material in this section is reproduced from the previous work of Lewis [8]

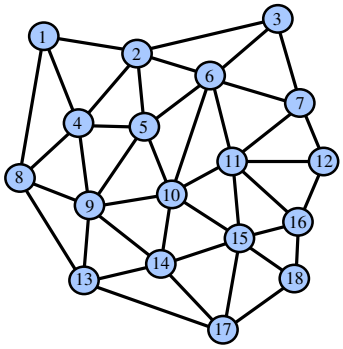
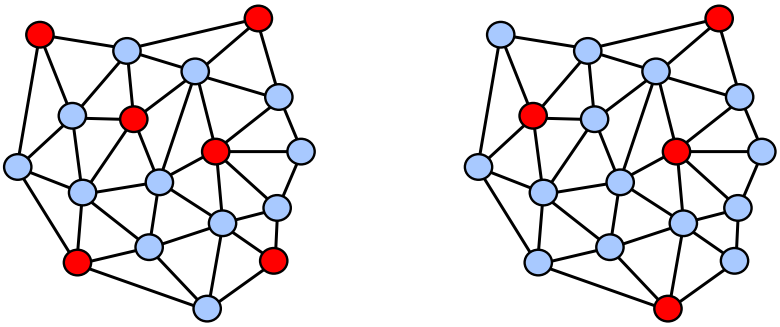
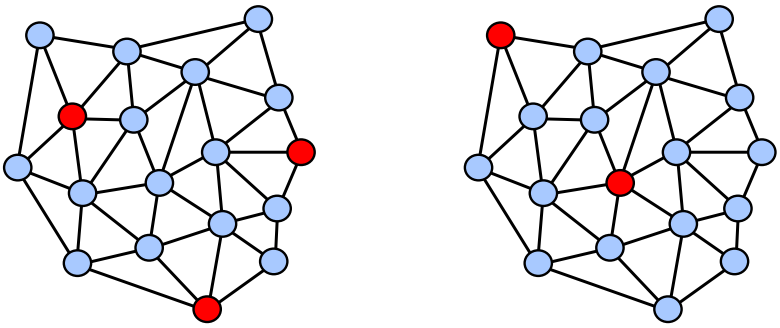


Figure 2.1. The simple graph used for illustration, which is the nodal graph of a triangular mesh.



(a) (b)

Figure 2.2. Examples of two different MIS(1) in the same graph. Vertices in the MIS are colored red. Note that the MIS(1) on the left contains six nodes, while that on the left contains only four.



(a) (b)

Figure 2.3. Examples of two different MIS(2) in the same graph. Vertices in the MIS are colored red. Note that the 2-MIS on the left contains three nodes, while that on the left contains only two.

makes it naturally suited for handling complex geometries, as the elements in the mesh can be chosen to conform to the boundaries of the geometry. Many disciplines, such as continuum mechanics, biophysics, and fluid dynamics, utilize FEM to simulate physical phenomena. The general FEM pipeline consists of three main tasks, the computation of the elemental local operator matrices, assembly of the local operator matrices into a global system of linear equations, and solving the linear system to determine approximation of the unknown function (see Fu et al. [4]). This work is focused on the linear solve step of the FEM pipeline, but that it occurs in the context of FEM is occasionally significant.[2]

2.4 Multigrid Methods

Here we describe the geometric multigrid method, which is a well-established method to solve linear systems associated with regular grids or lattices. We then discuss the algebraic multigrid method, an adaptation of the geometric multigrid method to solve linear systems arising from irregular domains, or meshes. We focus specifically on the smoothed aggregation multigrid variant of algebraic multigrid, the method used in this paper, and how it is commonly used as a preconditioner for other linear solvers.

Consider a sparse linear system, $A_0 x_0 = b_0$, which is a discrete approximation of $Dx = b$, where D is a differential operator and x, b are functions, sampled at a set of points Ω_0. Iterative techniques such as the Jacobi method, Gauss-Seidel, and successive over-relaxation exhibit fast convergence during the early iterations, which slows dramatically in the later stages. This can be easily observed by plotting the residual, $r_h = A_h \tilde{x}_h^k - b_h$, where $\tilde{x}_h^k$ is the approximate solution after k iterations. The stalled convergence in the later stages occurs because, while these techniques are effective at removing high-frequency error, they are inefficient at reducing "smooth" or low-frequency errors.

Multigrid methods are based on the idea that low-frequency components of error in a fine discretization appear as higher frequency components in a coarser

[2]The material in this section is reproduced from the previous work of Lewis [8]

discretization, where they can effectively be removed by iterative methods. Let A_1 be some approximation of A_0 defined at points Ω_1, $|\Omega_0| > |\Omega_1|$, R_0^1 be a restrictor which transfers vectors from Ω_0 to Ω_1, and P_1^0 be a prolongator which transfers vectors from Ω_1 to Ω_0. For an approximate solution $\tilde{x}_0$ of $A_0 x_0 = b_0$, the error $e_0 = \tilde{x} - x$ satisfies $A_0 e_0 = r_0$, and can be approximated by $\tilde{e}_0 = P_1^0 e_1$, where $A_1 e_1 = R_0^1 r_0$, if A_1 on Ω_1 approximates A_0 sufficiently well, and R_1^0, P_0^1 accurately transfer vectors between levels. Multigrid methods accelerate convergence of an iterative method by applying corrections from coarser levels, $\tilde{x}_0 \leftarrow \tilde{x}_0 + \tilde{e}_0$, to the approximate solution on the finer level. At each level, an approximate solution to the linear system is found by applying a relaxation method, and the residual is restricted to the next coarser level to solve for the correction. A direct solver is used to solve the linear system at the coarsest level. Each level of the hierarchy efficiently removes different frequency components from the error.

In geometric multigrid methods, $\{\Omega_0, \Omega_1, ..., \Omega_n\}$ are chosen to be regular grids or lattices, usually with each coarser grid a subset of the points in the previous level. The regular geometry of the grids simplifies definition of the operators needed to construct the multigrid hierarchy. While geometric multigrid is highly effective, it is limited to solving problems on regular domains. For unstructured domains, *algebraic* multigrid (AMG) methods [10] were developed in order to accelerate linear solvers for systems arising from unstructured meshes. As the name indicates, the coarse grid hierarchy is constructed using only algebraic properties of the linear system itself, and does not explicitly use the geometry of the underlying mesh. Many strategies exist for constructing the coarse grid hierarchy in algebraic multigrid methods. Generally, some compromise must be made between finding AMG hierarchies which approximate error very accurately, and quickly computing a reasonable one [12].

The smoothed aggregation multigrid method is described in detail in [12], [14], and [2]. Consider the graph G_0 with nodes corresponding to degrees of freedom in A_0, and edges corresponding to non-zero values in A_0. The coarse levels of the multigrid hierarchy are constructed by partitioning G_0 into disjoint aggregates, which are internally connected. The goal is to obtain an aggregation with fairly uniform

sizes and shapes of aggregates. A tentative prolongator $\tilde{P}$ is constructed as a matrix with a row for each degree of freedom and a column for each aggregate:

$$\tilde{P}_{ij} = \begin{cases} 1 & \text{if } i^{th} \text{ degree of freedom in } j^{th} \text{ aggregate,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

The final prolongator P is obtained by applying a smoothing operator to $\tilde{P}$, for example, damped Jacobi:

$$P = (I - \omega D^{-1} A)\tilde{P}. \tag{2.2}$$

The restrictor is then taken to be $R = P^T$ and the coarse level operator as $A_c = P^T AP$ [13].

In this work, we mainly consider the aggregation portion of the smoothed aggregation multigrid method. Since the definition of P, R, and A_c depend only on the choice of aggregation, it follows that how well $\tilde{e} = Pe_c$, where $A_c e_c = Rr$, approximates the error $e = \tilde{x} - x$ in $Ax = b$ also depends entirely on the aggregation chosen.

The AMG method can also be used as a preconditioner in Krylov subspace solvers such as the linear conjugate gradient, biconjugate gradient, and generalized minimum residual solvers [11]. It has been observed that the efficiency of PCG-AMG is greater than PCG with other preconditioners or the AMG method alone. It is heuristically explained in [12] that this is because AMG preconditioners try to eliminate all error components when compared with other preconditioners. Also, with the AMG method, some very specific error components may not be computed easily, or at all, due to imperfect interpolation from the solution of the coarser linear system, but Krylov subspace solvers are not affected by such issues.[3]

[3]The material in this section is reproduced from the previous work of Lewis [8]

CHAPTER 3

AGGREGATION METHODS

We use the term aggregation to refer to a labeling of the nodes of a graph such that all nodes with the same label form a connected subgraph, all graph nodes are labeled, and the values of labels are integers from one to the number of distinct labels. In this chapter, we review several algorithms that are employed to compute an MIS(k) and to aggregate nodes of a graph through the use of the computed MIS(k).

3.1 General Process

All MIS-based aggregation methods presented in this paper are similar in their general structure, which can be described as follows:

1. Select k such that the number of nodes obtained from the MIS(k) generation is roughly equal to the number of aggregates required. Use one of the algorithms described below to generate an MIS(k) of the graph.

2. Assign non-MIS(k) nodes to the aggregate rooted by the nearest MIS node. The distance is defined as the length of a shortest path between two nodes. There may be nodes that are equidistant from multiple root nodes, and one of many strategies may be used for breaking ties.

3. Apply conditioning operations to improve qualities (size etc.) of the aggregation or to enforce constraints that the initial aggregation does not meet.

Below, the MIS(k) algorithms in Step 1 are described in Section 3.2, the initial aggregation in Step 2 is described in Section 3.3, and the conditiong in Step 3 is described in Section 3.4.

3.2 MIS(k) Algorithms

For any given graph, there may exist many sets that are maximally independent, and these sets may have different cardinalities. We observed that the cardinality of MIS(k) used for aggregation has an effect on the quality of the resulting aggregation. In general, for k fixed, an MIS(k) of higher cardinality produces aggregations that perform better when used for AMG coarsening. We discuss three algorithms for finding an MIS(k) of a graph. In these algorithms, the nodes are classified as *in* if they are in the MIS(k) being constructed, *out* if they cannot be in the MIS(k) being constructed (due to conflicts with other nodes), and $free$ otherwise. Initially, all nodes are marked as $free$.

1. Lexicographic.

   The nodes of the graph are examined in a sequential order. If a node is marked as $free$ when being examined, it is marked as *in*, and all nodes with a shortest path of k or less are marked as *out*. Thus, the set produced is uniquely determined by the indexing of the graph nodes.

2. Flood-Fill.

   Initially, a node is chosen arbitrarily and marked as *in*, and all nodes with a shortest path of k or less to it are marked as *out*. Then, the set of all $free$ nodes with a shortest path of distance $k+1$ to an *in* node is found. The node in this set with the most adjacent nodes marked as *out* is chosen and marked as *in*, and all nodes with a shortest path of k or less to it are marked as *out*. The process is repeated until all nodes are marked as either *in* or *out*.

3. Randomized Parallel.

   This method is a variation of that proposed by Luby [9]. All $free$ nodes are assigned a positive integer value at random. All *in* and *out* nodes are assigned the value zero. All $free$ nodes with a value greater than the value of each node with a shortest path less than or equal to k from it are marked as *in*, and the nodes with a shortest path less than or equal to k from them are marked as *out*, i.e., each unassigned node is added to the MIS(k) if it has a value greater than

all nodes in its k-neighborhood. This process repeats until all nodes are marked as *in* or *out*.

The lexicographic algorithm and the flood-fill algorithm are both difficult to parallelize effectively; the lexicographic algorithm is faster than the flood-fill algorithm, but the flood-fill algorithm usually produces significantly larger MIS(k). The randomized parallel version is easily implemented in parallel and its GPU implementation is faster than the CPU version of the flood-fill or lexicographic algorithm. It usually produces an MIS(k) of cardinality that is in between that of the flood-fill and Lexicographic algorithms. For a more detailed description of these algorithms and comparison of them, see Lewis [8].

3.3 Aggregation

An MIS(k) of the input graph is used to allocate graph nodes to aggregates. The MIS(k) nodes are each assigned to its own sequentially numbered aggregate, becoming the root nodes of the aggregates. The remaining nodes are then allocated to the aggregate of the closest root node. This is accomplished iteratively by having each unassigned node check its neighbors to see if they have been allocated to an aggregate. If an unassigned node has allocated neighbors, which are all allocated to the same aggregate, the node allocates itself to the same aggregate. If an unassigned node has allocated neighbors, which do not all belong to the same aggregate, the node is allocated to the aggregate to which it has the most neighbors belonging. The process repeats until all nodes have been assigned to an aggregate.

We chose to break ties using connectivity in order to preference aggregates with higher internal connectivity, which should result in reduced communication costs in the PCG-AMG solver for improved efficiency.

3.4 Conditioning

Our motivation for conditioning the aggregations was to make them suitable for use in a GPU-based AMG preconditioner. The primary constraint for this application is the size of the aggregates. Evenly sized aggregates result in a balanced workload for each GPU thread block, improving efficiency. Only a limited amount of resources

can be allocated to a GPU thread block; this places a limit on the maximum size of an aggregate.

The sizes of aggregates produced with MIS(k) methods vary with k, as higher values of k result in fewer root nodes. In general, the aggregate size distribution is a normal distribution, and the vast majority of aggregates are very close in size to the average size, but there are a few outlying aggregates whose sizes significantly differ from the average size (see Lewis [8]). Since there are relatively few outliers, it is reasonable to devise heuristics for improving the size regularity.

In Fu et al. [4], conditioning by exchanging nodes between adjacent aggregates was introduced. Since it only transfers a single node from or to each aggregate, it could take a large number of cycles to correct a single aggregate, which was much too large. While this approach was effective in eliminating very large aggregates, in Lewis [8], it was found to produce aggregations with disconnected partitions, which are invalid for AMG coarsening, when tighter size constraints were specified. Checking to ensure a node exchange will not disconnect an aggregate is expensive to implement on the GPU, and makes the method unnacceptably slow.

To more efficiently enforce size constraints, we present a conditioning method which uses the following operations:

1. Two adjacent aggregates may merge into one.

2. An aggregate may split into two new aggregates.

3. Two adjacent aggregates may first merge into one, and then split into two (Merge-Splitting).

Merging two aggregates is implemented by simply assigning all nodes from one aggregate into the other, and renumbering the aggregates.

Splitting is implemented by finding the pair of nodes, within the aggregate, with the longest path distance between them. These nodes become root points for the new aggregates. The remaining nodes are assigned one at a time, with each new aggregate alternately being assigned the unassigned node closest to its root point. This results in

an as even a split as possible. Splitting an aggregate is accomplished by a single GPU thread block using shared memory to store the adjacency of the aggregate subgraph.

The merging and splitting operations affect the size distribution more significantly than exchange of single nodes, so fewer operations need to be applied to achieve the same results.

Merging, splitting, and merge-splitting operations are used in the fixed ratio methods detailed below, which condition aggregations to have a specified average aggregate size (coarsening ratio), and for all aggregates to be within a specified size range.

3.5 Implementations of Methods

We have implemented and tested the following methods for aggregation in the AMG coarsening phase:

1. **Fixed Ratio GPU** produces an initial aggregation using an MIS(k) produced with the randomized method, and conditions it to enforce the specified coarsening ratio, and size constraints.

2. **Fixed Ratio CPU** is a CPU implementation of the GPU method described above, using an MIS(k) produced with the flood-fill method.

In both methods, it is necessary to find the most desirable merges, splits, and merge-splits that are possible. We define a desirability rating of merges, splits, and merge-splits, which gives the highest score to the operations that reduce the standard deviation of the part sizes by the greatest amount. Given the mean size of aggregates, μ, two adjacent aggregates A and B, and notation $|X|$ to denote the number of nodes contained in the aggregate X, we define the desirability of merging of A and B as:

$$D_M(A,B) \equiv (|A| + |B| - \mu)^2 - (|A| - \mu)^2 + (|B| - \mu)^2, \tag{3.1}$$

and the desirability of A and B merge-splitting as:

$$D_{MS}(A,B) \equiv 2((|A| + |B|)/2 - \mu)^2 - (|A| - \mu)^2 + (|B| - \mu)^2. \tag{3.2}$$

To achieve the specified coarsening ratio, the number of aggregates the aggregation should have is found; if the current aggregation should have more aggregates the

most desirable splits are performed until it has the appropriate number of aggregates; similarly, if it should have fewer aggregates, the most desirable merges are performed until it has the appropriate number of aggregates. Then, the most desirable merge-splits are performed until the aggregation meets the specified size range.

CHAPTER 4

TESTING METHODOLOGY

4.1 FEM Solver

For testing the performance of the aggregation methods for multigrid coarsening, we used the FEM application described in [4] as a testbed. The application solves the elliptic Helmholtz equation over an irregular domain using a preconditioned conjugate gradient solver with an algebraic multigrid preconditioner in order to solve the resulting linear system from the discretized form of the PDE and the unstructured mesh. As the smoothing steps in the multigrid method can be performed independently, multiple iterations of smoothing are performed on partitions of the elements without global synchronization. The smoothing step on each of these partitions is handled by an individual GPU thread block. For the solver to efficiently use available resources, these partitions must be large enough that each block does as much work as possible, while not being so large as to require more resources than are available to a block. In order to accommodate this requirement, the aggregation phase of the multigrid setup must not only aggregate the fine mesh for the next AMG level, but also provide a partitioning of the fine-level aggregates into groups containing many aggregates. Each partition must contain all the nodes of all the aggregates present in the partition. We will refer to these two levels of aggregation as the "fine aggregation" and the "coarse aggregation" of a level, hereafter. There are two strategies for producing such a two-level aggregation: "bottom-up" and "top-down". In the "bottom-up" approach, the fine aggregation is created first, the graph induced by the fine aggregation is aggregated next, and the aggregation projected onto the input graph to form the coarse aggregation. The "top-down" approach is to create the coarse aggregation by partitioning the input graph first. Then, each partition of the input graph is partitioned individually to create fine aggregates. The MIS-based methods are naturally suited for

the bottom-up approach. Metis, as a general purpose graph partitioning library, can be used in either fashion. In our preliminary numerical experiments, we found that the use of the top-down approach had a very negative effect on the iteration count of the solver, as well as the solution time. In order to have an equitable comparison of the suitability of the different aggregation methods for AMG coarsening, we use the "bottom-up" approach for the Metis-based aggregator. Since only the fine aggregation determines the AMG level coarsening, using the same approach for coarse aggregations for both methods is appropriate for comparing their effectiveness as a coarsener for AMG.

4.2 Adaptations to MeTis

In our experiments, we used MeTis version 4.0.3. Since MeTis is designed for graph partitioning, but not for aggregation in the AMG method, there were two issues observed: a) large memory allocation when large number of partitions were specified and b) disconnected or empty partitions. When specifying more than about 20,000 partitions, MeTis fails because it attempts to allocate more memory than is possible. To correct for this issue, when more than 16,000 partitions are needed, we first call MeTis to partition the graph into four parts, and then, for each resulting subgraph, call Metis to partion into one-fourth the number of partitions originally required. Disconnected partitions were treated as separate aggregates and empty partitions were removed. The time taken for these postprocessing steps was not recorded in our timing results.

4.3 Metrics of Aggregation Quality

Graph partitions are typically optimized for the number of edges crossing from one partition to another while holding the number of nodes within a partition to be close to the mean. The edge-cut is defined as the ratio of the number of edges that have end points in different partitions to the number of edges that have both end points in the same partition. Graph partitioning is carried out in the context of load balancing for parallel scientific computing. The edge-cut corresponds to the volume of communication in such applications. Thus, the edge-cut is a good metric for such

purposes. For AMG aggregations, as we shall demonstrate further in the next section, it is not an ideal metric to improve the performance. In this section, we describe the following three novel topology-informed metrics for measuring the quality of an aggregation: convexity, eccentricity, and minimum enclosing ball (MEB) metrics.

As described in the introduction, MIS-based aggregations have aggregates which appear more regularly shaped and round than those of MeTis aggregations. Our hypothesis is that the difference in AMG performance seen for the different aggregation methods is due to the difference in the shape of the aggregates and that quantifying the shape of an aggregate would lead to a metric which correlates with the performance of the PCG-AMG solver. Shape is a geometric concept, and in a continuous space, the ideal shape of our aggregate is a sphere (our hypothesis), which is convex and has zero eccentricity. The metrics we define apply these geometric concepts to sets of vertices in a graph. Our metrics intuitively measure to what extent an aggregate differs from a sphere. Further, we also define a metric to measure the quality of a set of aggregates when the quality of the individual aggregate in the set are given, i.e., we combine the metric values for each aggregate to produce an overall score for an aggregation.

Consider a graph $G(V, E)$ composed of a set of vertices V and edges E, where each edge consists of a unordered pair of vertices, (e_1, e_2). An aggregation of G is a collection of sets of vertices $a = \{a_1, a_2, ..., a_n\}$ such that all vertices in V are in one of the sets and the sets are disjoint, i.e.,

$$\bigcup_{i=0}^{n} a_i = V \quad \text{and} \quad \bigcap_{i=0}^{n} a_i = \varnothing. \tag{4.1}$$

A quality metric for an aggregation is a function $m(G, a)$, which takes a graph and an aggregation of the graph as input and returns a scalar value. We denote the length of the shortest path distance between vertices v_1 and v_2 ($v_1, v_2 \in G$) as $p(G, v_1, v_2)$ and use $B_r(v_1)$ to denote the set of all nodes v_i such that $p(G, v_1, v_i) \leq r$. For a set of vertices $s \subset V$, we use G_s to mean the subgraph of G consisting of the nodes contained in s and the set of edges $\{(e_i, e_j) \in E : e_i \in s \text{ and } e_j \in s\}$, i.e., the edges of G which have both endpoints contained in s.

4.3.1 Convexity

A set of points in a continuous space is convex if the shortest line connecting any pair of points lies within the set. The graph analog of a straight line connecting two nodes is a shortest path between them. Whereas a line in a continuous space is unique, there may exist multiple distinct shortest paths between two nodes in a graph. We define a set of nodes as convex if for every pair of nodes in the set there exists a shortest path consisting only of the nodes in the same set. Thus, $s \subset V$ is convex if $\forall v_i, v_j \in s$, $p(G_s, v_i, v_j) \in G_s$. An aggregate a_i in G may or may not be convex. In order to measure the convexity of an aggregate, we define the aggregate convexity score of a_i as $|a_i|/|c|$, where $c \subset V$ is the smallest convex set which contains a_i. In practice, finding c may require checking all possible combinations of nodes in V which contain a_i as a subset. Since it is not feasible to combinatorially explore all possibilities, we use a heuristic algorithm to find a minimal convex set containing a_i, which we use as an approximation of c.

The algorithm finds a minimal convex set containing a_i by solving the boolean satisfaction problem: Let $paths(s, e)$ be the set of all shortest paths between nodes s and e, and $ext(p)$ be the set of node indices of all nodes in path p not contained in a_i. Let n_i be a boolean variable which is true if node i is added to a_i. The cost of n_i is 1 if true and zero if false. Then the algorithm finds a minimum cost satisfying assignment of:

$$\prod_{s,e \in a_i} \sum_{p \in paths(s,e)} \prod_{e \in ext(p)} n_e. \tag{4.2}$$

The nodes forming the minimum assignment are added to a_i and the process is repeated until no nodes are added. The algorithm terminates, and a_i is convex.

4.3.2 Eccentricity

An ellipsoid in a continuous space has eccentricity equal to $\sqrt{1 - a^2/b^2}$, where a is the maximum distance from the centroid of the ellipsoid to the surface and b is the minimum distance from the centroid to the surface. For a graph aggregate, we define the centroid of a set of vertices s as $\{x \subset s : \sum_{v \in s} p(G_s, x, v) = \min_{v_i \in s} \sum_{v \in s} p(G_s, v_i, v)\}$, i.e., the set of vertices in s where the sum of all shortest paths from the vertex to all

others in s is minimum. The centroid c may not be a single vertex. Thus, we define the path distance $p(G, c, v)$ to be the average of all $p(G, c_i, v)$, $c_i \in c$. A vertex $v \in a_i$ is on the boundary of a_i if v is adjacent to a vertex that is not in a_i. We define the eccentricity score of an aggregate a_i (with c_i being the smallest convex set containing a_i) as the ratio of the minimum distance from the centroid of c_i to a boundary node of c_i divided by the maximum distance from the centroid of c_i to a boundary node of c_i.

4.3.3 Minimum Enclosing Ball

A sphere of radius r centered at a point c in a continuous space is the set of all points whose distance to c is less than or equal to r. The analogue in a graph is the ball $B_r(v)$ around vertex v, which is the set of all nodes in the graph with path distance to v less than or equal to r. An object in continuous space approaches a sphere as its volume approaches that of the smallest sphere that fully contains it. The size of the minimum enclosing ball (MEB) of an aggregate a_i, $minBall(a_i)$ is $\min_{x \in V} |B_r(x)|$ such that $a_i \subset B_r(x)$. We define the aggregate MEB metric as $|a_i|/minBall(a_i)$ and compute the aggregation MEB metric as the arithmetic mean of all aggregate MEB scores. This metric measures, in a sense, how far from being a sphere an aggregate is.

CHAPTER 5

EXPERIMENTAL RESULTS

In this chapter, we give details about how the experiments were run and the meshes used, and then present the results from our numerical experiments. We examine the correlation between our quality metrics and solution time, as well as iterations. We examine the correlation between our quality metrics and the solution time (the time required to numerically solve the linear system that arises from the mesh and discretization of a PDE, with AMG levels and operators defined) to verify if higher quality aggregations are helpful in obtaining the solution more efficiently (shorter amount of time). We also examine the computational context in which the MIS-based techniques provide higher quality aggregations than MeTis-based techniques.

For our numerical experiments, we used four meshes: blob mesh, brain mesh, unstructured mesh on a cube, and a structured mesh on a cube. All the meshes are composed of tetrahedral elements. These meshes were chosen in order to have a variety of domains. Table 5 provides further details. Our experiments were carried out on a Pentium Xeon X5650 (2.67GHz) server with 12GB of main memory that is equipped with an NVidia Tesla C2070 compute unit. The Tesla C2070 unit has 448 CUDA cores and 6 GB memory.

Table 5.1. Statistics for all meshes used: The node count, element count, minimum vertex valance (Min), and maximum vertex valance (Max).

Mesh	Nodes	Elements	Min	Max
Blobs	277,657	1,650,105	5	46
Brain	322,497	1,805,242	6	34
Structured	274,625	1,572,864	3	18
Unstructured	197, 561	1,122,304	3	25

Each experiment consists of running the FEM Solver, specifying the aggregation method, coarsening ratio, and mesh. The initial values used were fixed for each mesh over all experiments. We record the elapsed time for performing the aggregations required to construct the AMG hierarchy (aggregation time), and the elapsed time for performing the linear system solve using the constructed AMG hierarchy (solution time). We also record the number of iterations required for the solver to converge to the solution (iteration count), the Convexity, Eccentricity, and MEB results for each aggregate in every aggregation, and the edge-cut ratio for every aggregation.

We consider coarsening ratios in $\{15, 20, 25, 30, 35\}$, and for each combination of aggregation method, mesh, and coarsening ratio, repeat the experiment 20 times, reporting average value over the 20 iterations for all results. Since the MIS aggregation methods may produce different aggregations given the same input in different test runs, there is more variation seen in the results than for MeTis.

We present results for solution time, aggregation time, and iteration count in Figure 5.1. In many cases, we see that MeTis-based aggregations take significantly more iterations than MIS-based aggregations, while when the MeTis-based aggregations take fewer iterations, the margin is usually smaller. This possibly indicates that the preconditioner is not as efficient due to the aggregation technique. We clearly see that MeTis takes the longest time for aggregation time, followed by MIS-CPU method and then the MIS-GPU method. It is also notable that aggregation time dominates solution time by a wide margin, and that in all cases, the MIS-based methods are significantly faster than the MeTis-based methods for aggregation, for both the GPU and serial implementations. The fact that our algorithm can be implemented on a GPU makes it an attractive option for aggregation.

For each aggregate metric, we compute the arithmetic mean, median, first quartile, and last quartile of the aggregate metrics in each aggregation, as candidate metrics on an aggregation. We then compute the Pearson product moment correlation coefficient of each candidate metric with solution time in Figure 5.2 and with iteration count in Figure 5.3. We also compute the Pearson product moment correlation coefficient of edge-cut with solution time and with iteration count and include it in the plots for

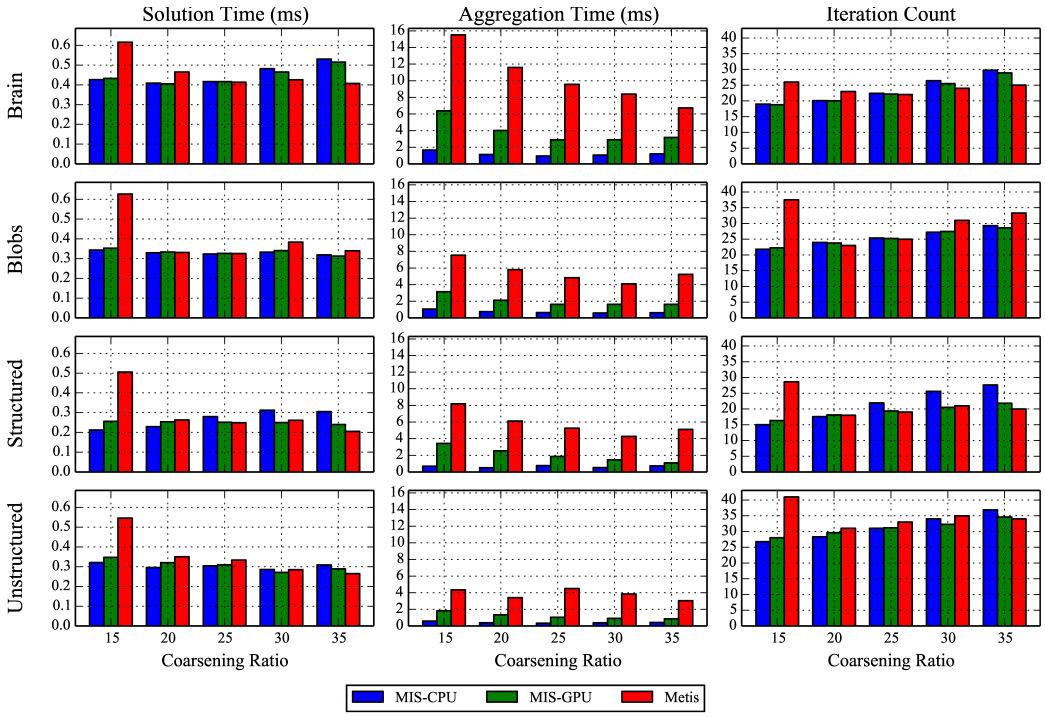


Figure 5.1. Solution time, aggregation time, and iteration count results. Each subplot shows the average value by coarsening ratio and aggregation method. The labels along the top indicate the metric score shown in that column, the labels on the left show the mesh used, bar color denotes the aggregation method used, and bars are positioned by coarsening ratio.

comparison.

Clearly, the convexity first quartile aggregation metric is most correlated with both solution time and iteration count and shows significantly more correlation with AMG performance than does edge-cut. All of the metrics introduced show more correlation with solution time than does edge-cut; however, the eccentricity metrics are less correlated with iteration count.

It is clear that the strongest predictor of AMG performance is the convexity first quartile metric. AMG performance seems to be more affected by small amounts of aggregates with very low scores than by large amounts of aggregates with very high scores. This is indicated by the low correlation of the convexity third quartile metric with iteration count and with solution time, relative to the other convexity metrics.

In Figure 5.4, we show a scatter plot of convexity first quartile metric scores against iteration count over all our experimental results. The aggregation method used is

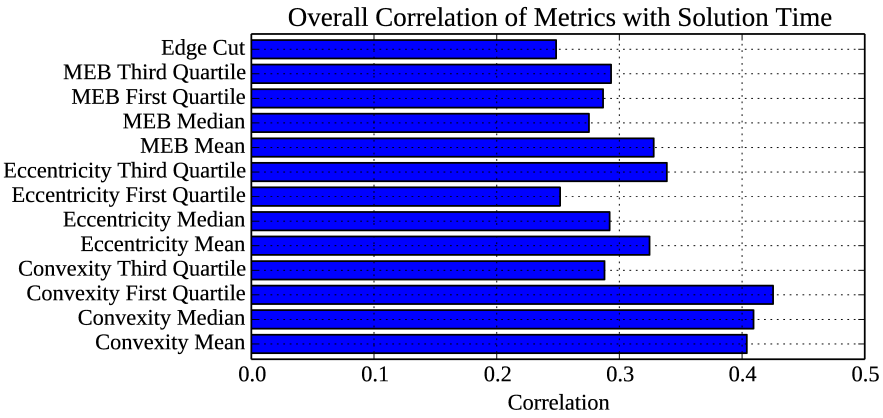


Figure 5.2. Pearson correlation coefficient of each metric with solution time

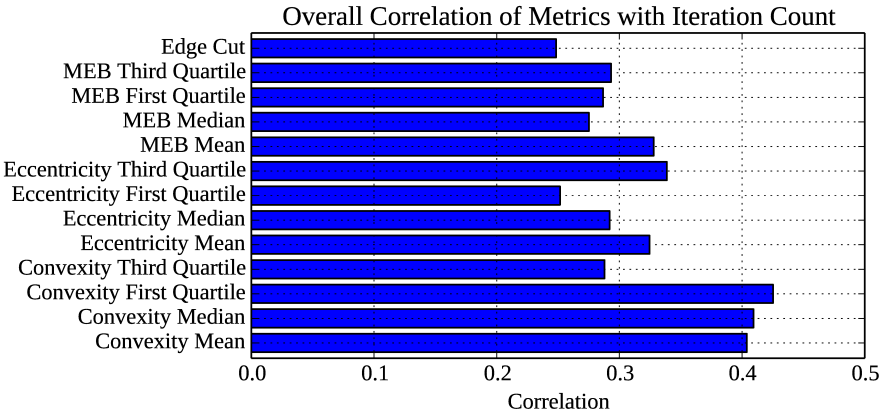


Figure 5.3. Pearson correlation coefficient of each metric with iteration count

indicated by the marker shape, the coarsening ratio by the marker size, and the mesh by marker color. The markers are rendered as mostly transparent so the amount of overlap can be seen. Figure 5.5 shows a similar plot for convexity first quartile metric scores against solution time. The scatter plots show visible correlation, with the results for each mesh observed seperately showing higher correlation than all results together. In general, a poorer convexity metric indicates that the solution time for AMG-PCG linear solver is high. This trend also holds for each mesh taken indidually, i.e., for circles of the same color, the correlation is high. We should add that this trend is absent when we control for both the mesh and coarsening ratio. We believe

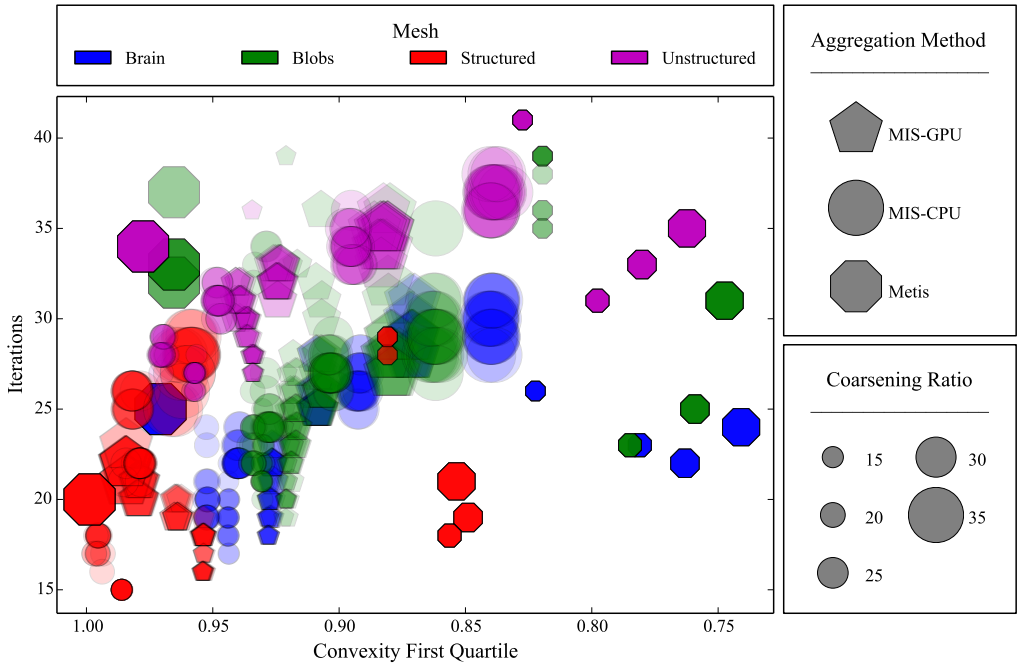


Figure 5.4. Scatter plot of the convexity first quartile metric against the iteration count

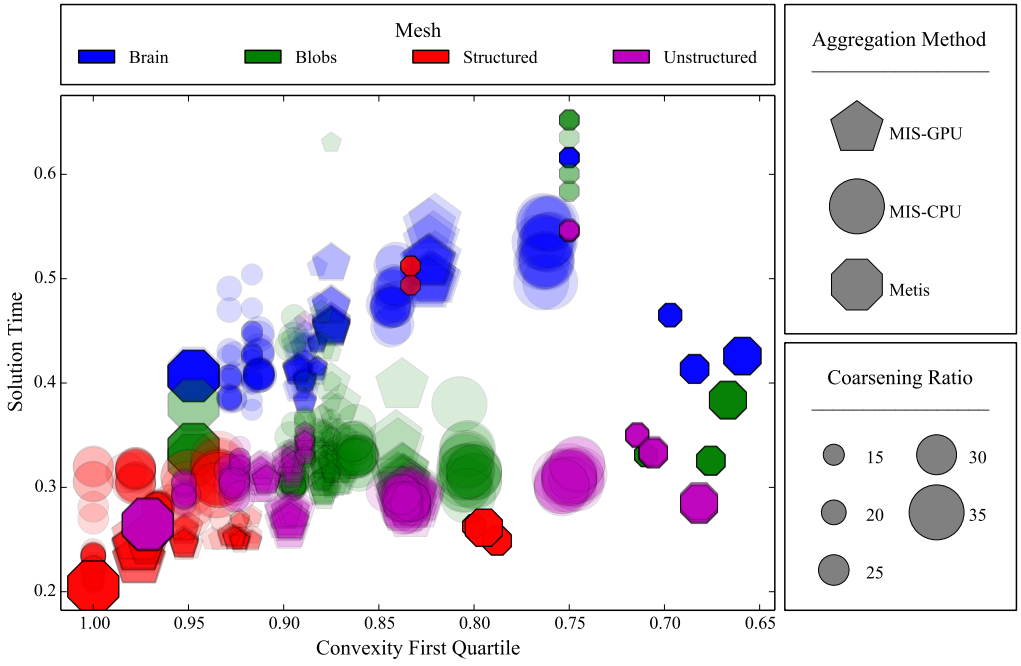


Figure 5.5. Scatter plot of the convexity first quartile metric against the solution time

that the reason for the absence of the trend is due to lack of sufficient data points. It is very difficult to obtain many aggregations whose quality vary enough to observe a trend when we are restricted to use only a handful of techniques. Thus, we leave the examination of the correlation for constant coarsening ratios as future work.

We visualize the quality of aggregations produced by each of the techniques using a cumulative distribution plot of the convexity aggregate metric scores, shown in Figure 5.6. Each of the figures plot the normalized number of aggregates in an aggregation whose convexity metric is poorer than a certain value. Just as any

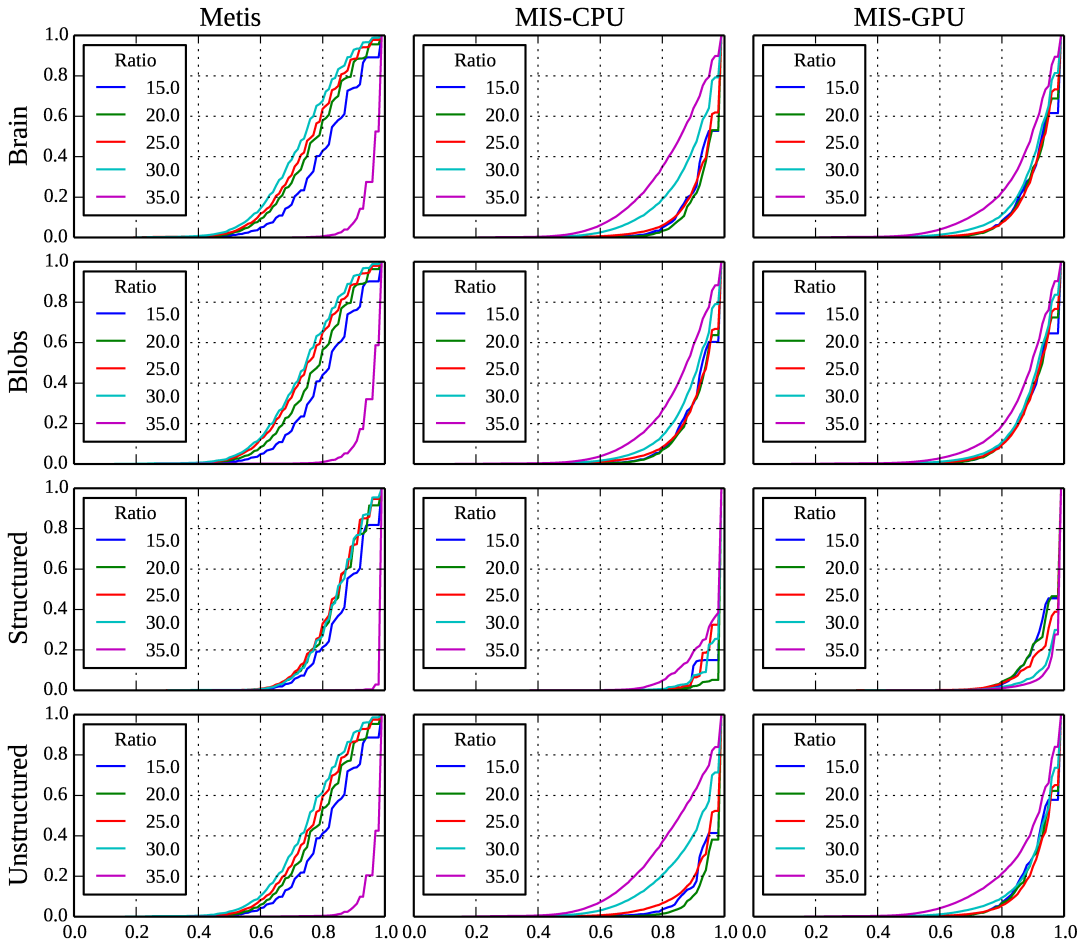


Figure 5.6. Cumulative distribution of aggregate convexity metric scores. Each curve corresponds to a coarsening ratio. The plot is similar to a cumulative probability distribution curve, i.e., each point indicates the percentage of aggregates whose quality is below a certain value. The row and column labels indicate the mesh and aggregation method, respectively, of the plots.

cumulative probability distribution plot, it is a nondecreasing curve. In the figures, the cumulative aggregate distribution is plotted for each combination of aggregation method and mesh, and separate lines, within each plot, indicate the coarsening ratio.

Visually, if the area under the curve is greater, there are more poor quality aggregates. It can be seen in the figures that the quality of the aggregation is poorer for the MeTis-based aggregation technique, i.e., there are more aggregations with poorer quality. For the MeTis-based aggregations, we see that the quality of aggregations generally decreases from coarsening ratio of 10 to 30. For a coarsening ratio of 35, however, the quality of aggregations drastically improves to the best value. This may be due to an effect of the recursive partitioning algorithm employed by MeTis. For a large coarsening ratio, since the number of partitions is small, MeTis may be as effective as, or more effective than, the bottom-up MIS-based techniques at producing high-quality aggregations. For the MIS-based CPU and GPU algorithms, the quality of aggregates is generally the best when the coarsening ratio is around 20-25. This is because the MIS computation naturally provides a coarsening ratio of around 23 without conditioning. As the effect of the conditioning steps to achieve the desired coarsening ratio are not significant, the quality of aggregations produced by an application of MIS algorithms is very good.

In Figure 5.7 we show the metric scores for each metric, for each combination of mesh, aggregation method, and coarsening ratio. For lower coarsening ratios, the convexity of the aggregations are generally best for the MIS-CPU-based technique, followed by the MIS-GPU-based technique, and finally the MeTis-based aggregation technique. For the highest coarsening ratios used, MeTis-based aggregations have the lowest solution time. This is because the quality of aggregates is better, for larger coarsening ratios, with MeTis-based techniques. For smaller coarsening ratios, MIS-based techniques produce better quality aggregates and result in lower solution time. This also indicates that the conditioning steps significantly affect the quality of the aggregates. We leave the development of quality-aware conditioning techniques as future work.

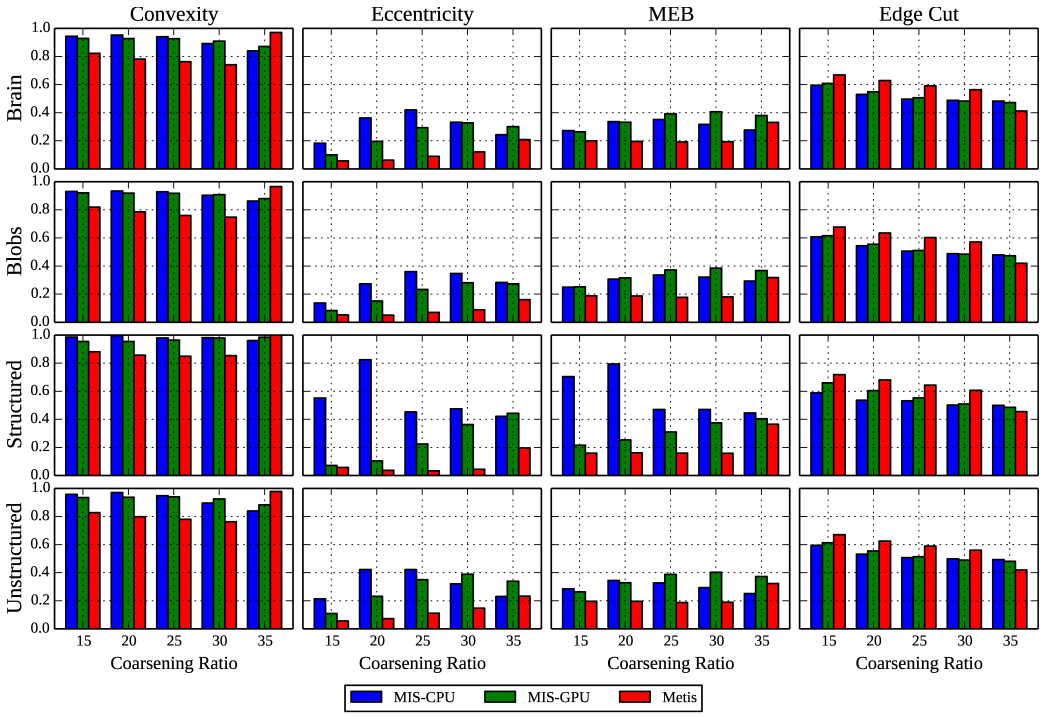


Figure 5.7. Comparison of metric scores. Each subplot shows the average metric score by coarsening ratio and aggregation method. The labels along the top indicate the metric score shown in that column, the labels on the left show the mesh used, bar color denotes the aggregation method used, and bars are positioned by coarsening ratio. All plots are at the same scale.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

In this thesis, we discussed two main approaches for aggregation of degrees of freedom for AMG-preconditioned Krylov subspace-based linear solvers: (a) the top-down approach and (b) the bottom-up approach. We used the Metis graph partitioner for the top-down approach, and we used MIS-based aggregators for the bottom-up approach. We compared the relative merits and demerits of the two approaches in a series of numerical experiments. We designed several metrics that were used to evaluate the quality of the aggregations. These metrics were designed based on our hypothesis that a "good" aggregate is roughly convex and spherical in shape. We also solved the Helmholtz equation using the PCG-AMG solver and reported the time it took to compute the solution. Based on the quality of aggregations and the solution time, we found that the top-down approach is suitable when the number of aggregations required is small, i.e., when the coarsening ratio required by the AMG preconditioner is large. On the other hand, the bottom-up approach is suitable when the number of partitions required is large, i.e., when the coarsening ratio is small. In general, the bottom-up MIS-based CPU and GPU techniques produced better quality aggregates when measured using the convexity metric that we designed. For large coarsening ratios, the top-down Metis-based technique was able to produce higher-quality aggregations. We also found that the solution time and the convexity of the aggregation had a reasonably good correlation. The correlation of the solution time with other metrics, however, was not strong.

In our MIS-based technique, we have used conditioning steps to generate aggregations of the desired size from the initial aggregation by merging and splitting aggregates and exchange of nodes from one aggregate to another. Our heuristic algorithm does not take the convexity metric into account. A possible future research direction is to

develop both CPU and GPU algorithms that take these metrics into account. Our metrics are only a function of the topology of the mesh, but the geometry of the aggregates may also be important for some applications. More research is needed to evaluate the quality of aggregates for solving other PDEs where the geometry may play an important role. There are applications where anisotropic meshes may be needed to solve the problem. In such cases, the topology-based metrics may need to be modified to account for the anisotropy in the geometry. Future research is needed to answer such questions as well. Finally, a top-down approach that optimizes the convexity metric may need to be developed for use in AMG preconditioners. Current techniques optimize the number of edge-cuts between aggregates. We hope that our paper influences research into many of these open questions.

REFERENCES

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

[2] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Mathematics of computation, 31 (1977), pp. 333–390.

[3] C. M. Fiduccia and R. M. Mattheyses, *A linear-time heuristic for improving network partitions*, in Proceedings of the 19th Design Automation Conference, DAC '82, 1982, pp. 175–181.

[4] Z. Fu, T. J. Lewis, R. M. Kirby, and R. T. Whitaker, *Architecting the finite element method pipeline for the {GPU}*, Journal of Computational and Applied Mathematics, 257 (2014), pp. 195 – 211.

[5] L. Hagen and A. Kahng, *New spectral methods for ratio cut partitioning and clustering*, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 11 (1992), pp. 1074–1085.

[6] G. Karypis and V. Kumar, *MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 5.0*. `http://www.cs.umn.edu/~metis`, 2009.

[7] B. Kernighan and S. Lin, *An Efficient Heuristic Procedure for Partitioning Graphs*, The Bell Systems Technical Journal, 49 (1970).

[8] T. J. Lewis, *Graph aggregation with k-maximal independent sets implemented on gpu for multigrid coarsening*, bachelors thesis, University of Utah, 201 Presidents Cir, Salt Lake City, UT 84112, 2014.

[9] M. Luby, *A simple parallel algorithm for the maximal independent set problem*, in Proceedings of the seventeenth annual ACM Symposium on Theory of Computing, 1985.

[10] J. Ruge and K. Stüben, *Efficient solution of finite difference and finite element equations by algebraic multigrid*, in Multigrid Methods for Integral and Differential Equations, vol. 3 of The Institute of Mathematics and its Applications Conference Series, Oxford, Clarendon Press, 1985, pp. 169–212.

[11] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd ed., 2003.

[12] K. Stüben, *A review of algebraic multigrid*, Journal of Computational and Applied Mathematics, 128 (2001), pp. 281–309.

[13] R. S. Tuminaro and C. Tong, *Parallel smoothed aggregation multigrid: Aggregation strategies on massively parallel machines*, in Proceedings of the 2000 ACM/IEEE Conference on Supercomputing, Supercomputing '00, Washington, DC, USA, 2000, IEEE Computer Society.

[14] P. Van.k, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing, 56 (1996), pp. 179–196.