A GPU-Based MIS Aggregation Strategy: Algorithms, Comparisons, and Applications Within AMG

T. James Lewis, Shankar P. Sastry, Robert M. Kirby and Ross T. Whitaker
Scientific Computing and Imaging Institute
University of Utah
Salt Lake City, UT, USA
Email addresses: {tjlewis, sastry, kirby, whitaker}@sci.utah.edu

***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. We present a GPU implementation of a "bottom-up" partitioning scheme based on maximal independent sets (MIS). We also present some novel topology-informed metrics that measure the quality of a partition. To test our implementation and the metrics, we use an existing AMG preconditioned conjugate gradient (PCG-AMG) solver and show that our metrics are correlated with the time and the number of iterations needed for the linear system to converge to a solution. For comparable coarsening ratios, we show that the MIS-based aggregation methods outperform Metis-based "top-down" aggregation method for the PCG-AMG method. Our results also indicate that MIS-based aggregation methods provide aggregates that are evaluated more favorably by our metrics than the aggregates provided by the Metis-based method.**

***Index Terms*—multigrid; GPU; maximal independent set; aggregation;**

I. Introduction

Relaxation techniques used to solve sparse linear systems are able to reduce high-frequency errors in the solution within a few iterations. For removing lower-frequency errors, these techniques are rather inefficient, in part motivating the development of multigrid methods. The Algebraic Multigrid (AMG) method is a robust iterative method that creates a series of graphs of increasing coarseness (with the finest level being the induced graph from the original matrix) in order to efficiently remove lower-frequency errors that are accrued in each level. The AMG method provides fast convergence by reducing low-frequency errors efficiently by applying coarse corrections and removing high-frequency errors by fine smoothing, with different frequencies of error being most effectively removed in different levels of the graph hierarchy. It has been observed that the AMG method is very efficient when it is used as a preconditioner for the linear conjugate gradient solver [1] and other Krylov subspace methods. The AMG method is explained in more detail in Section 2.

The preconditioned conjugate gradient [2] solver with AMG preconditioner (PCG-AMG) is typically used for numerically solving partial difference equations (PDEs) using the finite element method (FEM). The degrees of freedom of the solution in such a problem correspond to the solution at the vertices of a finite element mesh. One of the critical steps in the AMG method is the coarsening phase in which the degrees of freedom (mesh vertices in the finest level) are aggregated together to construct a coarser graph that can efficiently and accurately compute the lower-frequency errors.

The performance of the PCG-AMG solver depends on the quality of the aggregations that are constructed from the finer graphs. The aggregations have to be coarse enough to reduce the cost of the computation of the solution and must also be fine enough to resolve the errors that have been observed in the finer graph. Therefore, the size of aggregates ideally would capture the lower-frequency errors efficiently. Since the aggregates are obtained from a geometric embedding (unstructured mesh) of a graph, the shape of the aggregates also plays a significant role in the performance of the solver. The AMG method, however, is oblivious to the underlying geometry. Thus, a measure of the quality of an aggregation must be obtained from only the topology of the graph at a fine level.

In this paper, we address the following questions: a) how do we 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? In order to answer these questions, we employ two kinds of techniques to aggregate the graph nodes: a) a Maximal independent set (MIS)-based "bottom-up" method and b) a Metis-based "top-down" method. The aggregation methods are described in Section 3. We use the FEM and PCG-AMG solvers with the aggregators to solve the elliptic Helmholtz equation and compare the solution times. We carry out the numerical experiments on four meshes. The testing methodology and description of our aggregation quality

metrics are provided in Section 4. In practical applications, note that the linear solver may be used multiple times, so 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 a correlation is present among the aggregate quality, solution time, and the total number of iterations. These results are provided in Section 5. We conclude the paper and indicate future research directions in Section 6.

II. Background and Related Work

In this section, we provide some background on the concepts and prior work that are related to techniques developed in this paper.

A. Metis

Metis [3] is a graph partitioning software 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 [4], 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 [5], the Fiduccia-Mattheyses method [6], or other methods. We consider this a "top-down" approach because the partitioning phase occurs at the coarsest level and then is projected back 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.

B. The Algebraic Multigrid Method

In this section, we describe the algebraic multigrid (AMG) method in some detail. The AMG method is a well-established method to solve linear systems. Consider a sparse linear system, $Ax = b$, that is used to solve a PDE on a lattice. Any appropriate iterative method can be used to solve such linear systems. Some techniques such as the Jacobi method, Gauss-Seidel, and successive relaxation exhibit fast convergence during the early iterations and slower convergence in the later stages. In the early stages, the high-frequency errors are efficiently eliminated by these iterative techniques. This can be easily observed by plotting the residual, $r = b - Ax^k$, where x^k is the solution after k iterations, on the grid. The stalled convergence in the later stages occurs because the techniques are inefficient at reducing "smooth" or low-frequency errors. A multigrid method is based on the idea that the low-frequency errors, e, can be computed more efficiently on a coarser grid, and these errors can be interpolated back to the finer grid. In other words, $Ae = r$ can be solved on a coarse grid, and the solution to the linear equation can be obtained by adding e to x^k, i.e., $x^* = x^k + e$, where x^* is the solution of the linear system. For a large system obtained from large grids, the lower-frequency errors are hierarchically eliminated by a repeated application of the multigrid method until the resulting systems are small enough to be solved efficiently with a direct technique.

For unstructured meshes, however, the hierarchy of coarse meshes is harder to obtain. Thus, *algebraic* multigrid methods [7] were developed in order to accelerate linear solvers for unstructured meshes. As the name indicates, only the matrix, but not the geometry of the unstructured mesh, is taken into account in the AMG method. In this paper, we mainly consider the coarsening phase of the AMG method. The degrees of freedom in our linear system, which correspond to mesh vertices at the finest level, are aggregated together in the coarsening phase to construct a coarser graph with fewer degrees of freedom. A linear system is then solved using the coarse aggregation in order to efficiently compute the low-frequency errors, and the solution is interpolated to the finer level.

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 [2]. 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 [1] 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 due to imperfect interpolation from the solution of the coarser linear system, but the iterative solvers account for such issues.

C. Smoothed Aggregation Multigrid Technique

The smoothed aggregation multigrid method is described in detail in [2], [8], and [9]. The main step in the interpolation step (from the coarse graph to fine graph) in this method is the construction of aggregation and prolongator. The aggregation is constructed by choosing a set of graph nodes as root points and grouping its neighbors into one aggregate. If there are still unassigned nodes, they can be grouped with their nearest aggregate, or they form a separate aggregate if they are next to each other. The prolongator is an operator, such as the damped Jacobi smoother [10], that is used to interpolate the solution. The goal of these steps is to have a fairly uniform sizes and shapes of aggregates. As described in [10], MIS-based methods can be used to construct the aggregates. In this paper, we implement the GPU-based MIS aggregation algorithm for PCG-AMG.

III. Aggregation Methods

An MIS of a graph is a set of nodes such that no pair of nodes in the set is connected by an edge (independence), and it satisfies the property that addition of any nodes violates the independence property (maximality). Maximal independent sets may be generalized by including a parameter for the radius of independence [11]. Thus, an MIS(k) of a graph is a set of nodes such that no pair of nodes in the set has a shortest path of length less than k between them and is maximal. 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 section, we present 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).

A. 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 B, and the grouping and conditioning in Step 2 and 3 are described in Section C.

B. 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, 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), or $free$ 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. If the ordering of the nodes is such that every node has a degree less than or equal to the following node, this algorithm produces a "lexicographic MIS" of the graph.
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 [12]. 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) aggregations. It usually produces MIS(k) of cardinality that is in between that of the flood fill and lexicographic algorithms. The randomized parallel version is easily implemented in parallel, and its GPU implementation is faster than the CPU version of the flood fill and lexicographic algorithms.

C. Aggregation and Conditioning

An MIS(k) of the input graph is used to allocate graph nodes to aggregates. The in nodes, which are also the MIS(k) nodes, become the root nodes for aggregates. The non-MIS(k) are allocated to their closest root node. This is accomplished by having each unassigned node check its adjacent nodes to see if they have been allocated to an existing partition. If a neighbor is allocated to an aggregate, then the node may allocate itself to the same aggregate. When a node is equidistant from two root nodes, one of the following techniques may be used to break the tie. The first option is to break the tie arbitrarily. In our implementation, the node is allocated to the first acceptable aggregate. Since we are operating under the intuition that aggregates with high internal connectivity are likely to be more efficient for the PCG-AMG solver, we also tried breaking the ties using the connectivity information. When a node may be allocated to more than one aggregate, it is allocated to the aggregate to which the largest number of the node's neighbors are allocated. Another option is to consider the current size of each potential aggregate and assign the node to the smallest one among them with the goal of improving the

size regularity. We did not pursue this option as our focus was on finding effective *parallel* methods, and it would be difficult to implement this heuristic in parallel; we leave it as possible future work.

Our motivation for conditioning the aggregations was to make them suitable for use in a GPU-based AMG preconditioner. The primary constraint in this application is in regard to the size of the aggregates. Aggregates need to be evenly sized so that the workload is well distributed, and no aggregate should be so large that it requires more resources than that can be allocated to a GPU thread block. The sizes of aggregates produced with MIS(k) methods varies with k. We find that a good value of k is 2. 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. Since there are relatively few outliers, it is reasonable to devise heuristics for improving the size regularity. The heuristics carry out the following operations:

1) Aggregates may exchange nodes with their neighboring aggregates.
2) Two adjacent aggregates may merge into one.
3) An aggregate may split into two new aggregates.
4) Two adjacent aggregates may first merge into one, and then split into two (Merge-Splitting).

The exchange of nodes between aggregates works reasonably well when a few aggregates that larger than the maximum allowable aggregate size. In such cases, the aggregates that are too large are very likely to be adjacent to aggregates that are not as large, and by transferring a node from the larger aggregate to the smaller aggregate, size regularity can be achieved. The merging and splitting operations affect the size distribution more significantly than the exchange of single nodes. These operations do not result in discontinuous aggregates. The removal of a node from an aggregate may cause the aggregate to become disconnected, and, therefore, the aggregate may become invalid for the AMG preconditioner. It is very expensive to check for this condition on a GPU. Thus, we do not perform this operation. Effective conditioning methods allow the development of the fixed ratio methods detailed below, where the only parameter required is the coarsening ratio desired. The internal parameter, k, for the MIS(k) can be hidden from external users.

D. Implementations of Methods

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

1) **Fixed Ratio GPU** produces an aggregation with the specified coarsening ratio and applies a conditioning operation to improve the size distribution of the aggregates.
2) **Fixed Ratio CPU** is a CPU implementation of the GPU method described above.

In both methods, it is necessary to find the most desirable merges, splits, and merge-splits. 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:

$$D_M(A,B) \equiv (|A|+|B|-\mu)^2 - (|A|-\mu)^2 + (|B|-\mu)^2,$$

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

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

The Fixed Ratio GPU aggregator uses the randomized algorithm to produce an MIS(k), and then allocates nodes to the aggregate of the nearest MIS(k) node (ties are broken by connectivity). Then, the initial aggregation is obtained, and the number of aggregates needed for the specified coarsening ratio is calculated. The conditioning operations are carried out in order to obtain size regularity and the desired coarsening ratio. The operations are prioritized based on the desirability rating, and they are carried out for all aggregates until their rating is below a certain threshold.

IV. Testing Methodology

A. FEM Solver

For testing the performance of the aggregation methods for multigrid coarsening, we used the FEM application described in [13] 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 aggreagates present in the partition. We will hereafter refer to these two levels of aggregation as the "fine aggregation" and the "coarse aggregation" of a level. 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 is 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

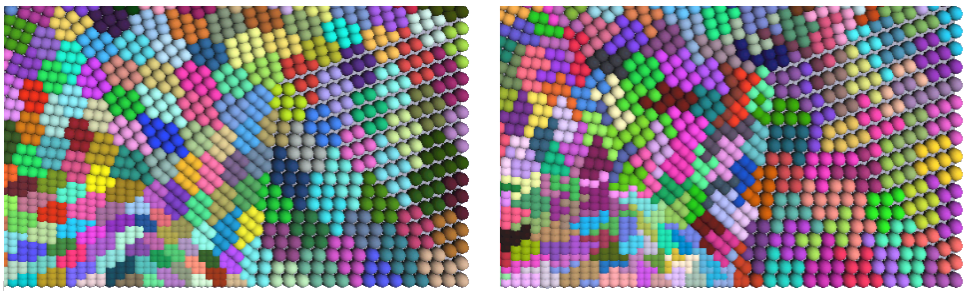

Fig. 1. An irregular triangular mesh aggregated with an MIS(2) aggregator (left) and with Metis (right). The nodal graph is visualized with colored balls representing nodes in the mesh; each color indicates an aggregate to which each node is assigned.

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.

B. 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 for a large number of partitions and b) disconnected or empty partitions. As Metis did not function properly due to a failed request for large memory allocation (for around 20,000 partitions), we split the graph into four parts (using Metis), and Metis was used to split each resulting subgraph into one-fourth the number of partitions originally required for the application. Disconnected partitions were treated as separate aggregates and empty partitions were ignored. The time taken for these postprocessing steps was not recorded in our timing results.

C. 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.

We first describe the motivation behind the development of the metrics. In our investigation, we empirically observed that an MIS-based aggregation scheme improved performance of our PCG-AMG solver significantly over the use of Metis-based aggregation schemes. When we visually compare an MIS aggregation with a Metis aggregation (see Fig. 1), we notice that the aggregates produced by the MIS-based method appear "rounder" than those produced by the Metis-based aggregation schemes. We believe that the difference in the performance is due to the difference in the shape of the aggregates and that quantifying the shape of an aggregate would lead to a metric that 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 is 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 an 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.*,

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

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 ($v_1, v_2 \in V$) 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 $\{(e_i, e_j) \in E : e_i \in s$

and $e_j \in s\}$, *i.e.*, the edges of G that have both endpoints contained in s.

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 analogue of a straight line connecting two nodes is the 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 that contains a_i. In practice, finding c may require checking all possible combinations of nodes in V that contain a_i as a subset. Since it is not feasible to combinatorially explore all possibilities, we use heuristic algorithm 1 to find an approximate c. The aggregation convexity metric is computed by taking the arithmetic mean of all aggregate convexity scores.

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. The aggregation eccentricity metric is computed as the arithmetic mean of the aggregate eccentricity scores of all the aggregates.

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. An object in continuous space approaches a sphere as its volume approaches that of the smallest sphere that fully contains it. The analogue in a graph is the ball $B_r(v)$ around vertex v. 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 a sphere an aggregate is.

Algorithm 1 Find minimal convex set containing a_i

```
1: procedure MakeConvex(a_i)
2:     c ← a_i
3:     x ← ∅
4:     repeat
5:         opt ← {∅}
6:         for all v_i, v_j ∈ a_i do
7:             good ← false
8:             paths ← GetPaths(v_i, v_j)
9:             for all path ∈ paths do
10:                if ∀x ∈ path : x ∈ c then
11:                    good ← true
12:                end if
13:            end for
14:            if good = false then
15:                for all path ∈ paths do
16:                    opt ← opt ∪ {{x ∈ path : x ∉ c}}
17:                end for
18:            end if
19:        end for
20:        x ← ⋃_{o∈opt} o
21:        for all s ⊂ x do
22:            good = false
23:            for all o ∈ opt do
24:                if o ⊄ x then
25:                    good ← false
26:                end if
27:            end for
28:            if good = true and |o| < |x| then
29:                x ← o
30:            end if
31:        end for
32:    until x = ∅
33: end procedure
34: procedure GetPaths(v_i, v_j)
35:    return {x ⊂ V : p(G_x, v_i, v_j) = |x| = p(G, v_i, v_j)}
36: end procedure
```

V. Experimental Results

In this section, we present the results from our numerical experiments. 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 generated four meshes: unstructured blob mesh, unstructured brain mesh, unstructured mesh on a cube, and a structured mesh on a cube. All the meshes contain tetrahedral elements. These meshes were chosen so that we have several types of domain and meshes. 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.

In our experiments, for each mesh, we varied the aggre-

gation methods and coarsening ratios. Since our MIS-based aggregation technique is initiated with a random seeding, we report the mean solution time for 20 executions of the FEM solver in which we solve the same linear equation. The variation in the solution time of the MIS-based is due to the variation in the aggregations produced by the technique. In general, they are slightly different for each execution.

		Ratio				
	Mesh	15	20	25	30	35
MIS-GPU	Blobs	0.344	0.330	0.324	0.333	0.318
	Brain	0.426	0.408	0.416	0.481	0.530
	Structured	0.212	0.229	0.280	0.312	0.305
	Unstructured	0.321	0.295	0.304	0.286	0.309
MIS-CPU	Blobs	0.353	0.334	0.327	0.340	0.312
	Brain	0.433	0.405	0.417	0.464	0.515
	Structured	0.256	0.254	0.251	0.250	0.240
	Unstructured	0.347	0.320	0.309	0.271	0.289
Metis	Blobs	0.627	0.331	0.325	0.384	0.340
	Brain	0.616	0.465	0.413	0.426	0.407
	Structured	0.506	0.263	0.249	0.261	0.205
	Unstructured	0.546	0.350	0.333	0.285	0.264

TABLE I
AVERAGE SOLUTION TIME

		Ratio				
	Mesh	15	20	25	30	35
MIS-GPU	Blobs	21.8	24.0	25.4	27.2	29.3
	Brain	19.0	20.1	22.4	26.4	29.8
	Structured	15.0	17.6	21.9	25.6	27.6
	Unstructured	26.8	28.3	31.0	34.0	36.8
MIS-CPU	Blobs	22.2	23.8	25.2	27.4	28.6
	Brain	18.8	20.0	22.2	25.5	28.9
	Structured	16.3	18.1	19.4	20.5	21.8
	Unstructured	28.0	29.6	31.2	32.3	34.6
Metis	Blobs	37.5	23.0	25.0	31.0	33.3
	Brain	26.0	23.0	22.0	24.0	25.0
	Structured	28.6	18.0	19.0	21.0	20.0
	Unstructured	41.0	31.0	33.0	35.0	34.0

TABLE II
AVERAGE PCG-AMG ITERATIONS

		Ratio				
	Mesh	15	20	25	30	35
MIS-GPU	Blobs	1.064	0.748	0.641	0.583	0.620
	Brain	1.665	1.131	0.954	1.056	1.198
	Structured	0.682	0.500	0.752	0.525	0.732
	Unstructured	0.587	0.376	0.335	0.375	0.427
MIS-CPU	Blobs	3.149	2.123	1.622	1.637	1.629
	Brain	6.377	4.019	2.891	2.899	3.177
	Structured	3.434	2.537	1.854	1.462	1.081
	Unstructured	1.824	1.343	1.044	0.914	0.864
Metis	Blobs	7.535	5.811	4.841	4.100	5.243
	Brain	15.508	11.603	9.570	8.393	6.723
	Structured	8.193	6.128	5.266	4.271	5.113
	Unstructured	4.341	3.405	4.494	3.844	3.035

TABLE III
AVERAGE AGGREGATION TIME

Some preliminary results are provided in Tables I, II, and III. In Table I, the average time to solve the linear equations by the PCG-AMG method is provided for each aggregation technique for each mesh. These are average solution times that are provided for the sake of completeness. The scatter plots (explained later in this section) provide more detailed results. Table II provides the average number of iterations needed by the PCG-AMG solver to converge to a solution. In many cases, we see that Metis-based aggregations take more iterations than MIS-based aggregations, which possibly indicates that the preconditioner is not as efficient due to the aggregation technique. Table III provides the average time taken by each aggregation technique to produce the aggregation. Here, we clearly see that the Metis takes the longest time followed by the MIS-CPU method and then the MIS-GPU method. The fact that our algorithm can be implemented on a GPU makes it an attractive option for aggregation.

We first present the results for the correlation between the average solution time and quality of an aggregation. In Fig. 2, the scatter plots of our results are provided for all the metrics considered in this paper. In each plot, the aggregation quality is plotted on the X-axis, and the solution time is plotted on the Y-axis. The colors of the scatter points (circles, pentagons, and octagons) indicate the domain on which the mesh was generated. The size of the shapes indicates the coarsening ratio; a larger circle indicates a larger coarsening ratio. Brighter shapes indicate that many data points lie close together. A circle indicates MIS-CPU was used for aggregation; a pentagon indicates MIS-GPU was used for aggregation; and an octagon indicates Metis was used for aggregation.

As can be seen in Fig. 2(a), the convexity metric has a reasonably good correlation with the solution time. 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, *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 that the reason for the absence of the trend is the lack of sufficient data points. It is very difficult to obtain many aggregations whose quality varies 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. For all other metrics, however, as shown in Fig. 2(b), (c), and (d), the correlation is not strong. This was expected for the edge-cut ratio metric as it was not developed for linear solvers but for load balancing purposes. These results indicate that the convexity of an aggregation is more important than other metrics. We also notice that Metis gives us some very nonconvex aggregations when the coarsening ratio is very small.

Our next set of results explores the quality of the aggregations provided by each of our techniques and their relationship with the solution time. The details can be seen in Fig. 3. For a lower coarsening ratio, the convexity of the aggregation is better for the MIS-CPU-based technique, followed by the MIS-GPU-based technique, and finally the Metis-based aggregation technique. For a higher coarsening ratio, Metis-based aggregations produce high-quality aggregations, which is also reflected

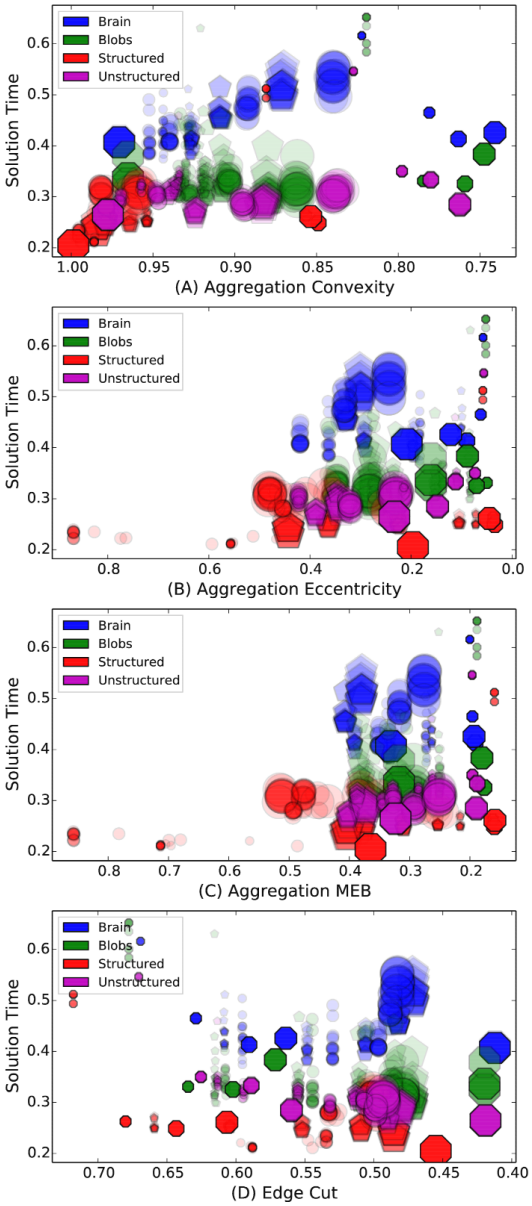

Fig. 2. Scatter plots for (a) aggregation convexity, (b) aggregation eccentricity, (c) aggregation MEB, and (d) edge cut metrics. The plots show the relationship between the metrics and the solution time. Each shape represents the results from an individual execution, the color of the shape indicates the mesh that was used, and the size corresponds to the coarsening ratio. Darker shapes indicate an overlap in data points. Circles indicate MIS-CPU was used to generate the aggregates; pentagons indicate MIS-GPU was used; and octogons indicate that Metis-based aggregators were used. These plots indicate that the solution time is more correlated with the aggregation convexity metric than with the other metrics, and MIS-based aggregators produce high-quality aggregations by the convexity metric.

in the solution time for the PCG-AMG linear solver. For the other two metrics (eccentricity and MEB), we do not see any such trends.

We visualize the quality of aggregations produced by each technique using cumulative distribution plots as shown in Fig. 4. Each figures plot the normalized number of aggregations whose quality is poorer than a certain value. Just as any

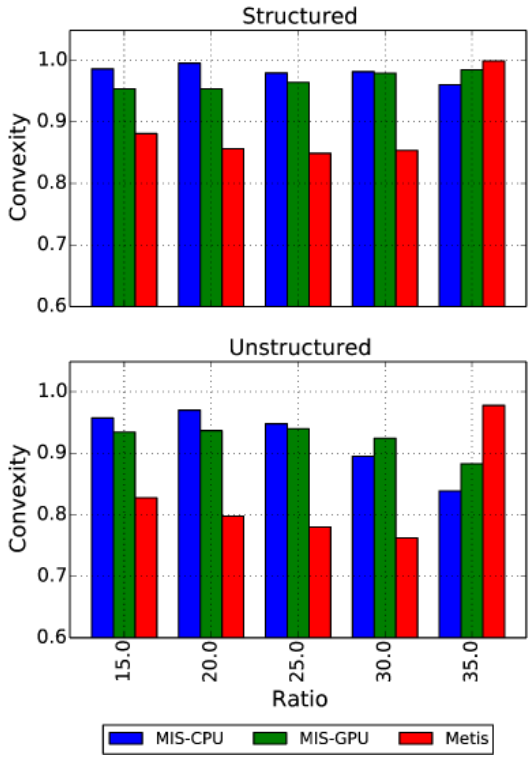

Fig. 3. The average convexity for the structured and unstructured meshes on the cube domain for various coarsening ratios. It can be seen that for a large coarsening ratio, the Metis-based aggregator yield higher-quality aggregations.

cumulative probability distribution plot, it is a non-decreasing curve. In the figures, the cumulative aggregation distribution is plotted for each coarsening ratio for each technique for each mesh. Since we found that only the convexity metric has a correlation with the solution time, we provide the plots only for the convexity metric.

Visually, if the area under the curve is greater, there are more poor quality aggregations. 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 a coarsening ratio of 10 to 30. For a coarsening ratio of 35, however, the quality of aggregations drastically improves to the best value, which may be due to an effect of the recursive partitioning algorithm employed by Metis. As the number of partitions increases, the top-down technique may not yield high-quality aggregations. 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 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 is not significant, the quality of aggregations produced by an

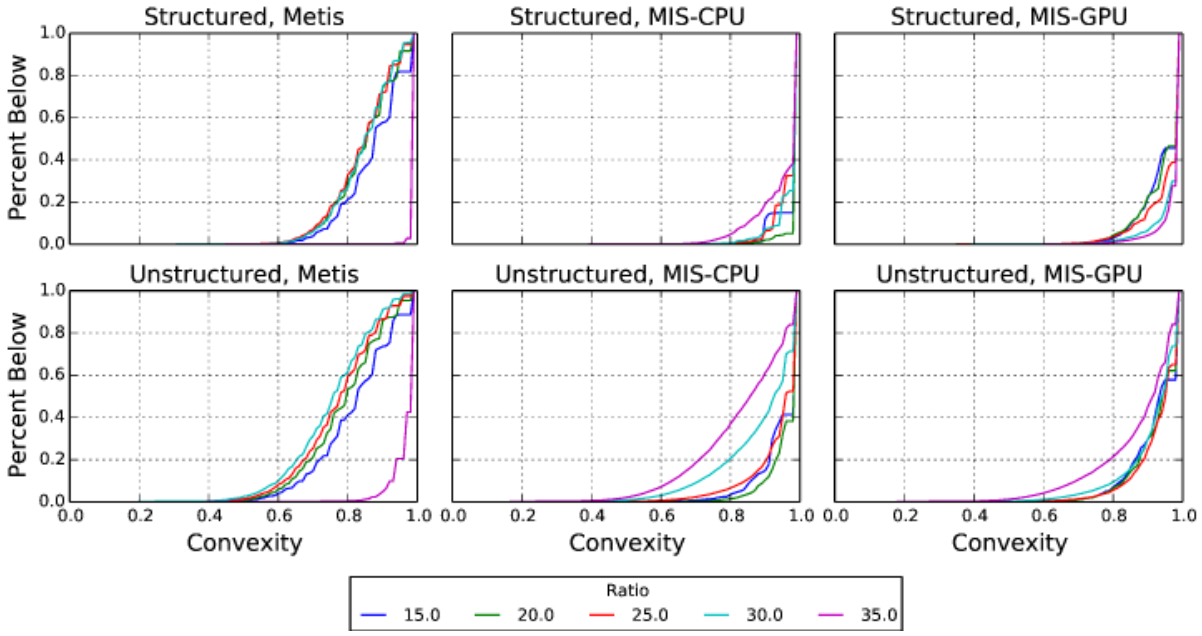

Fig. 4. The cumulative distribution of the individual aggregate convexity metric. Each curve corresponds to a coarsening ratio. The plot is similar to a cumulative probability distribution curve, *i.e.*, each point indicates the percentage of aggregates whose quality is below a certain value. It is a nondecreasing curve. The structured mesh is on the top row, and the unstructured mesh is in the bottom row. Each column corresponds to an aggregation technique; the left column corresponds to the Metis-based technique, and the two columns to the right correspond to the MIS-based technique.

application of MIS algorithms is very good.

In our experiments, we see that the time taken by Metis-based aggregations is lower than MIS-based methods for larger coarsening ratios because the quality of aggregates is better when Metis-based techniques are used for aggregations. For smaller coarsening ratios, MIS-based techniques produce better quality aggregates and result in lower solution time, which 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.

VI. Conclusions and Future Work

In this paper, 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.

ACKNOWLEDGMENTS

The work was supported by an NSF REU grant under NSFIIS-0914564, the NIH/NIGMS Center for Integrative Biomedical Computing grant 2P41 RR0112553-12, the DOE NET DE-EE0004449 grant, the DOE NET DE-EE0004449 grant, and ARO W911NF1210375 (Program Manager: Dr. Mike Coyle) grant. The authors would like to thank Zhisong Fu for his finite element code and the meshes used to solve the Helmholtz equation. The authors would also like to thanks Christine Pickett, an editor at the University of Utah, for finding numourous typos in one of the draft of the paper.

REFERENCES

[1] K. Stüben, "A review of algebraic multigrid," *Journal of Computational and Applied Mathematics*, vol. 128, pp. 281–309, 2001.

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

[3] G. Karypis and V. Kumar, "MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 5.0," http://www.cs.umn.edu/~metis, University of Minnesota, Minneapolis, MN, 2009.

[4] 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*, vol. 11, no. 9, pp. 1074–1085, 1992.

[5] B. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," *The Bell Systems Technical Journal*, vol. 49, no. 2, 1970.

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

[7] 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*, ser. The Institute of Mathematics and its Applications Conference Series, vol. 3, Oxford. Clarendon Press, 1985, pp. 169–212.

[8] P. Vanek, J. Mandel, and M. Brezina, "Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems," *Computing*, vol. 56, no. 3, pp. 179–196, 1996.

[9] A. Brandt, "Multi-level adaptive solutions to boundary-value problems," *J. Math. Comput.*, vol. 31, no. 138, pp. 333–390, 1977.

[10] R. Tuminaro and C. Tong, "Parallel smoothed aggregation multigrid : Aggregation strategies on massively parallel machines," in *Supercomputing, ACM/IEEE 2000 Conference*, 2000, pp. 5–5.

[11] N. Bell, S. Dalton, and L. N. Olson, "Exposing fine-grained parallelism in algebraic multigrid methods," *SIAM Journal on Scientific Computing*, vol. 34, pp. 123–152, 2012.

[12] M. Luby, "A simple parallel algorithm for the maximal independent set problem," in *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, 1985.

[13] 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*, vol. 257, pp. 195 – 211, 2014.

[14] N. Bell and M. Garland, "Cusp: Generic parallel algorithms for sparse matrix and graph computations," 2012, version 0.3.0. [Online]. Available: http://cusp-library.googlecode.com

[15] F. Pellegrini, "Scotch and libscotch 5.1 user's guide," http://gforge.inria.fr/docman/view.php/248/7104/scotch_user5.1.pdf, University de Bordeaux, 2008.

[16] R. S. Tuminaro, "Parallel smoothed aggregation multigrid: aggregation strategies on massively parallel machines," in *Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM)*, ser. Supercomputing '00. Washington, DC, USA: IEEE Computer Society, 2000.

[17] M. Adams, "Heuristics for the automatic construction of coarse grids in multigrid solvers for finite element problems in solid mechanics," University of California Berkeley, Tech. Rep., 1999.

[18] T. F. Chan, J. Xu, and L. Zikatanov, "An agglomeration multigrid method for unstructured grids," in *in Tenth International Conference on Domain Decomposition*, 1998, pp. 67–81.

[19] D. Talmor, "Well-spaced points for numerical methods," Ph.D. dissertation, Carnegie Mellon University, 1997.

[20] A. Godiyal, J. Hoberock, M. Garland, and J. C. Hart, "Rapid multipole graph drawing on the GPU," in *Proceedings of the 16th International Symposium on Graph Drawing*, I. G. Tollis and M. Patrignani, Eds., 2009, pp. 90–101. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-00219-9_10

[21] M. Adams, "A parallel maximal independent set algorithm," EECS Department, University of California, Berkeley, Tech. Rep. UCB/CSD-98-993, Jan 1998. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/1998/5560.html

[22] P. Gajer, M. T. Goodrich, and S. G. Kobourov, "A multi-dimensional approach to force-directed layouts of large graphs," in *Proceedings of the 8th International Symposium on Graph Drawing*, ser. GD '00, 2001, pp. 211–221.