Bucket Sorted Independent Sets

David M. Alber*

Abstract

The setup phase in algebraic multigrid is responsible for building a hierarchy of coarse grids, linear operators, and transfer operators needed by the solve phase. Several coarse grid selection algorithms have been developed, and of those, independent set-based methods are the most numerous. Each of these algorithms searches the vertex weights in a graph to select new C-points and often also traverse the graph to identify vertices whose weights are to be updated. In this paper, a new algorithm for conducting the search for new C-points is introduced. Using a bucket data structure to organize the vertices, the algorithm does not traverse the graph to identify new C-points. A new technique for updating vertex weights is also introduced to compliment the search algorithm.

1 Introduction

The classical algebraic multigrid (AMG) [9, 11, 4] iterative linear solver decomposes into a setup phase and a solve phase. The setup phase algorithms perform several tasks including coarse grid selection, building restriction and prolongation operators, and construction of the coarse level operators. The solve phase uses the components built during setup to implement a multigrid cycle.

Independent set algorithms are often the basis for coarse grid selection [9, 7, 6, 10, 1, 5, 2]. These graph algorithms share common design properties, such as a routine to search for vertices that will become coarse grid points (C-points). Additionally, these algorithms update the weights of vertices as the coarse grid selection proceeds.

In this paper, we present new theory and algorithms to decrease the computational cost associated with the search and weight update procedures used in coarse grid selection.

The remainder of this paper is organized as follows. Section 2 outlines two parallel coarse grid selection algorithms, CLJP and CLJP-c, which are the basis for this work. A discussion on the search used in CLJP and CLJP-c is given in Section 3. It is shown that the final coarse grid selected by these algorithms is insensitive to the order in which C-points are chosen. This fact allows development of new search algorithms without changing the output produced by CLJP and CLJP-c. A new technique for updating weights in the graph is also presented in this section. A new algorithm, called Bucket Sorted Independent Sets (BSIS), that uses these methods is developed. Section 4 compares the performance of BSIS with CLJP-c. Conclusions and future directions of this research are discussed in Section 5.

2 Coarse Grid Selection Algorithms

The developments presented in this paper are applied to the CLJP-c coarse grid selection algorithm [1]. In this section, concepts on coarse grid selection, CLJP [6], and CLJP-c are presented.

The weights used in coarsening algorithms discussed in this paper are based on concepts of *strength of connection* and *strong influence*. The set of vertices that vertex i strongly depends on is defined as

$$S_i = \left\{ j : j \neq i, -a_{ij} \geq \theta \max_{k \neq i}(-a_{ik}) \right\}, \tag{1}$$

*Siebel Center for Computer Science, University of Illinois at Urbana-Champaign, 201 North Goodwin Avenue, Urbana, IL 61801, U.S.A., email: alber@uiuc.edu

where a_{ij} is the entry in row i, column j of the linear system A. In practice, θ is often 0.25. The set of vertices that i strongly influences is the set of vertices that strongly depend on i: $S_i^T = \{j : i \in S_j\}$.

These definitions are used in the construction of the *strength matrix*, S. The strength matrix is a boolean matrix with the entry in the ith row and jth column defined as follows:

$$S_{ij} = \begin{cases} 1 \text{ if } i \text{ is strongly influenced by } j, \\ 0 \text{ otherwise.} \end{cases} \tag{2}$$

Coarsening algorithms use S to form the graph on which the algorithm is run. The weight update rules in these algorithms are based on both the strong influences and strong dependencies of a vertex. The S matrix is not necessarily symmetric for a symmetric A matrix, and this affects the implementation of coarse grid selection algorithms. This impact is discussed further in Section 3.

Note that the set of vertices that i strongly depends on (S_i) is the nonzero entries in row i of S. Similarly, the set of vertices that i strongly influences (S_i^T) is a list of nonzero entries in column i of the strength matrix.

2.1 Cleary-Luby-Jones-Plassmann

The Cleary-Luby-Jones-Plassmann (CLJP) algorithm [6] was developed as the first completely parallel coarse grid selection algorithm. Under certain conditions, it is capable of producing the same coarse grid independent of the data distribution and number of processors used. Two important contributions from the development of this algorithm were the application of a parallel independent set algorithm (based on [8]) to coarse grid selection and the definition of weight update heuristics to create a single-pass algorithm where previous coarsening algorithms required two-passes over the vertices.

The base weight for a vertex in CLJP is determined by the number of vertices influenced by that vertex. For example, the base weight for vertex i is $|S_i^T|$. This is used in Ruge-Stüben (RS) coarsening [9], but this weight makes it difficult to guarantee set independence while selecting more than one vertex per iteration because many vertices share the same weight. Selecting several vertices in one iteration requires additional information to avoid choosing adjacent vertices. CLJP adds a random number to the weight of each vertex to break these ties. The augmented weight of vertex i is $w_i = |S_i^T| + \text{rand}[0, 1)$. The random augmentation makes it unlikely any two adjacent vertices will have the same weight, allowing the algorithm to identify a set of vertices in each iteration that are guaranteed to form an independent set. Each new C-point i selected by CLJP satisfies the following at the time of selection:

$$w_i > w_j, \quad \forall j \in \mathcal{N}_i, \tag{3}$$

where $\mathcal{N}_i$, defined below, is the *neighborhood* of i.

Definition 1. *The* neighborhood *of a vertex i, denoted $\mathcal{N}_i$, is the set of vertices in $S_i \cup S_i^T$.*

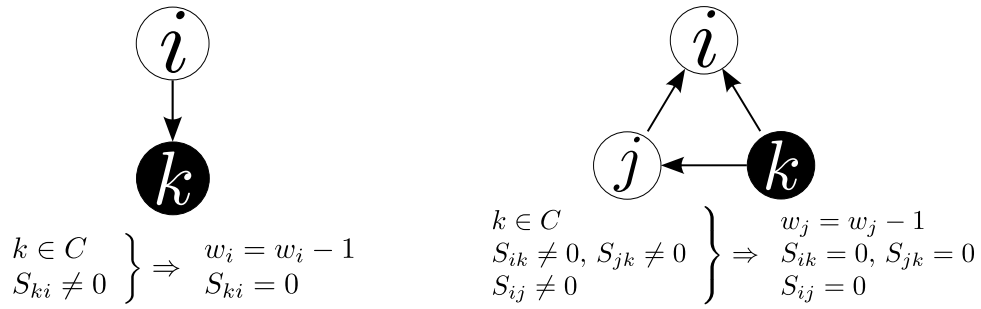
The weight update rules in CLJP remove edges from S and decrease the weight of a vertex, meaning that a vertex becomes less desirable as a C-point after its weight is updated. This happens in two cases.

1. Values at C-points are not interpolated; hence, a vertex that strongly influences a C-point is less valuable as a potential C-point itself. This case is shown in Figure 1(a).

2. If two vertices i and j strongly depend on a C-point and j strongly influences i, then j is less valuable as a potential C-point because the influence of j on i can be interpolated from the C-point. Figure 1(b) demonstrates this condition.

When a weight is updated, edges in the graph of S are removed. Equivalently, elements are removed from the sets of strong influencers and strong dependers. When CLJP removes all edges, coarse grid selection is complete.

CLJP combines these elements into the following algorithm.



(a) Rule 1

(b) Rule 2

Figure 1: Cases where vertex weight updates occur in CLJP following the assignment of vertex k to the C-point set. Edges in the graph of S are removed following a weight update.

1: $F = \emptyset$, $C = \emptyset$
2: **for all** $i \in V$ **do** $\{V$ is the vertex set of $S\}$
3: $\quad w_i \leftarrow |S_i^T| + \text{rand}[0, 1)$
4: **while** $|C| + |F| \neq |V|$ **do**
5: $\quad$ select independent set $D = \{i : w_i > w_j,\ \forall j \in \mathcal{N}_i\}$ (search step)
6: $\quad$ **for all** $j \in D$ **do**
7: $\quad\quad C = C \cup j$
8: $\quad\quad$ **for all** $k \in \mathcal{N}_j$ **do**
9: $\quad\quad\quad$ update w_k {see Section 2.3}
10: $\quad\quad\quad$ **if** $w_k = 0$ **then**
11: $\quad\quad\quad\quad F = F \cup k$

CLJP is a fully parallel coarse grid selection algorithm with several attractive features. The random numbers used to break ties, however, lead to poor performance compared to other parallel coarsening algorithms, such as Falgout coarsening (a CLJP-RS hybrid method) [7], on problems from structured grids and on problems in three dimensions. Adapting CLJP to be more competitive on such problems was the motivation for developing CLJP-c.

2.2 CLJP in Color

CLJP in color (CLJP-c) [1] was developed to improve CLJP performance on structured problems while retaining good properties of CLJP not shared by other algorithms. Structure in the coarse grids is provided by pre-coloring the graph of S and using information from the colors to augment the base weights. The weight for vertex i is the sum of $|S_i^T|$ and an augmentation derived from the graph coloring. The colors in the graph form a hierarchy, which breaks ties between vertices with the same base weight.

The use of graph coloring provides the following important result.

Theorem 1. *For all pairs of vertices i and $j \in \mathcal{N}_i$ CLJP-c guarantees $w_i \neq w_j$.*

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

This theorem demonstrates all adjacent vertices have different weights in CLJP-c. This is stronger than in CLJP, which is unlikely to have two adjacent vertices sharing the same weight, but provides no guarantee.

2.3 Search and Weight Update

In this section, the implementations and costs associated with searching and updating vertex weights in CLJP are discussed. The pseudo-code below assumes the software uses a compressed sparse row (CSR) [3] matrix format or other similar format, which are common matrix formats in numerical software. CSR provides low memory costs for storing sparse matrices and gives relatively quick access to the nonzeros in a row. Accessing the nonzeros in a column is an expensive operation in this format, and this expense strongly influences the weight update routine in CLJP.

CLJP and CLJP-c rely on a search routine to locate vertices with weights larger than their neighbors and on a weight update routine to modify weights of vertices connected to new C-points. Line 5 of the CLJP pseudo-code contains the search step, and Line 9 encapsulates the update step.

The search step in CLJP is implemented as follows.

```
1: for all i ∉ (C ∪ F) do
2:   if w_i > w_j, ∀j ∈ N_i then
3:     D = D ∪ i
```

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

Pseudo-code for the weight update in CLJP is shown below.

```
1: for all d ∈ D do
2:   for all i ∈ S_d do
3:     w_i ← w_i − 1 {see Figure 1(a)}
4:     S_d = S_d \ i {removing edge from graph}
5: for all i ∉ (C ∪ F) do
6:   for all k originally in S_i do
7:     if k ∈ D then
8:       mark k NEW-C-POINT
9:   for all j ∈ S_i do
10:    if j ∉ D then
11:      for all k ∈ S_j do
12:        if k is marked NEW-C-POINT then {i and j both influenced by same new C-point}
13:          w_j ← w_j − 1 {see Figure 1(b)}
14:          S_i = S_i \ j {remove edge from j to i}
15:  for all k originally in S_i do
16:    if k ∈ D then
17:      unmark k
18:      S_i = S_i \ k {remove edge from k to i, if present}
```

The level of complication in this update routine is due to the CSR format and the need to find vertices strongly influenced by new C-points. When a new C-point k is selected, the first type of weight update is trivial because k can cheaply determine which vertices influence it. The second type of update is more difficult because k is unable to easily determine, in a CSR-format matrix, the vertices it influences. Note that this update routine is more expensive than the search routine shown earlier. The update requires looking at many vertices i and all of their strong influencers j. Then the routine looks at the strong influencers of j to determine if any $k \in D$ strongly influences both i and j. The cost of doing this increases dramatically as the density of S increases. Large operator complexities have a disproportionately large impact on coarse grid selection run time. In the next section, we introduce a modified update routine to compliment the new searching technique.

3 Bucket Sorted Independent Set Selection

New techniques for searching the graph of S for new C-points and subsequently updating the weights of remaining vertices are developed in this section. We call the new algorithm Bucket Sorted Independent Sets (BSIS).

Like CLJP-c, BSIS depends on graph coloring, but unlike CLJP-c, it uses different routines for search and weight update. Furthermore, rather than applying the color information to augment vertex weights, BSIS uses the colors in a bucket data structure. Once initialized, this data structure selects independent sets, which satisfy the conditions in Equation 3, in constant time.

3.1 Coarse Grid Invariance

In CLJP-c, all vertices satisfying Equation 3 are selected in each iteration, followed by weight updates for neighbors of the new C-points. BSIS, on the other hand, does not select all vertices satisfying the conditions in each iteration. It selects vertices sharing the maximum weight in the graph and updates the weights of their neighbors. Despite this different approach to C-point selection, the coarse grids chosen in CLJP-c and BSIS are identical. In fact, any coarse grids selected satisfying Equation 3 will be identical. Theoretical results proving that statement are presented in this section.

The following two corollaries are a result of Theorem 1 and apply to CLJP-c and BSIS.

Corollary 1. *Any set of vertices sharing the same weight in the graph of S is an independent set.*

Corollary 2. *The set of vertices with the largest weight in the graph form an independent set satisfying Equation 3. That is, each vertex in that set has a uniquely maximal weight in its neighborhood.*

The first corollary states that independent sets can be selected in the graph simply by selecting sets of vertices with the same weight. Corollary 2 refines this observation to a subset of vertices guaranteed to satisfy the selection criterion. In particular, it shows it is possible to build the coarse grid by selecting vertices with the maximum weight, updating weights, selecting the next set of vertices with maximum weight, and so on.

Before demonstrating the invariance of the coarse grid, the following definitions are necessary.

Definition 2. *The* extended neighborhood *of a vertex i, denoted $\mathcal{N}_i^2$, is the union of the neighborhoods of the neighbors of i: $\mathcal{N}_i^2 = \bigcup_{j \in \mathcal{N}_i} \mathcal{N}_j$.*

The distinction of the extended neighborhood is important because only vertices in $\mathcal{N}_i^2$ affect the weights of i or its neighbors.

Definition 3. *Let D_k be the set of C-points selected by CLJP-c following the kth weight update pass. Therefore, D_0 is the set of C-points selected in the first iteration, D_1 is the set of C-points selected in the second iteration, and so on.*

Vertices in D_0 become C-points whether they are all selected simultaneously, followed by their associated weight updates (as in CLJP and CLJP-c), or whether subsets of the vertices are selected and weights are updated prior to the selection of another subset of eligible vertices. If vertex $j \in D_k$, $k > 0$ is selected before any vertex $i \in D_0$ (in a way that satisfies Equation 3), this has no effect on i becoming a C-point. This is proven in the following lemma.

Lemma 1. *Vertices in D_0 must become C-points irrespective of the order in which they are selected and the weights of their neighbors are updated.*

Proof. D_0 is an independent set, so weight updates resulting from making a vertex in D_0 a C-point does not affect any other vertex in D_0. All $i \in D_0$ satisfy $w_i > w_j$, $j \in \mathcal{N}_i$, and any weight updates affecting $j \in \mathcal{N}_i$ decrease w_j. Therefore, w_j can never become equal to or larger than w_i, proving vertex i must become a C-point. □

To prove that any method using the selection criterion in Equation 3 builds the same coarse grid, two points will be made: the vertices in all D sets become C-points in all cases and the same F-points are chosen by each method. The lemma below provides the framework to support both points.

Lemma 2. *A vertex $i \in D_k$ satisfies Equation 3 only after one or more vertices in $D_{k-1} \cap \mathcal{N}_i^2$ become C-points.*

Proof. The proof is by induction. The base case is D_1, which clearly satisfies the lemma statement. If a vertex in D_1 can be added to the coarse set before any vertex in D_0, then it is in D_0, not in D_1.

The inductive case states that no vertex in D_{k-1} can be added before one or more of the vertices in the intersection of its extended neighborhood and D_{k-2} are added.

The task is to demonstrate that before $i \in D_k$ becomes a C-point, one or more vertices in $\mathcal{N}_i^2 \cap D_{k-1}$ must become C-points. Assume (for contradiction) no vertices in $\mathcal{N}_i^2 \cap D_{k-1}$ have been added to the C-point set and i satisfies Equation 3.

Case 1: The set $\mathcal{N}_i^2 \cap D_{k-2}$ is empty. In this case, if i satisfies Equation 3, it is in D_l, $l < k-1$, rather than in D_k.

Case 2: The set $\mathcal{N}_i^2 \cap D_{k-2}$ is not empty. Two cases for the set of vertices $j \in \mathcal{N}_i^2 \cap D_{k-1}$ exist. First, if no $j \in \mathcal{N}_i \cap D_{k-1}$ exist then $i \cup D_{k-1}$ is an independent set and $i \in D_{k-1}$, contradicting the assumptions. Second, if there is one or more $j \in \mathcal{N}_i \cap D_{k-1}$, then it is known when all D_{k-2} are C-points that $w_j > w_i$. Based on the assumptions, $w_i > w_j$, implying one or more $k \in \mathcal{N}_i \cap D_{k-2}$ exists (because w_i must decrease in value for $w_j > w_i$ to become true, and this only happens when new C-points in $\mathcal{N}_i$ are assigned). This contradicts the assumptions because $w_k > w_i$ and i does not satisfy Equation 3. □

The result from Lemma 2 leads to the following corollaries.

Corollary 3. *A vertex $i \in D_k$ for any k always becomes a C-point.*

Corollary 4. *A vertex $i \notin D_k$ for any k will be an F-point in any selection scheme based on Equation 3.*

This is due to the update scheme. If the same set of C-points is always selected, then the weights of the other vertices will be the same and lead to an identical set of F-points.

These points combine to prove the invariance of the coarse grids.

Theorem 2. *The same coarse grid is selected by any method whose selection criterion is Equation 3.*

Proof. Lemma 2 and Corollary 3 prove vertices in D always become C-points. Corollary 4 states the set of F-points is identical for all methods. The same coarse grid is, therefore, always selected. □

Theorem 2 is an important result about the nature of coarse grids selected by CLJP-c. With this information, new algorithms can be created to yield identical coarse grids using different and possibly more efficient techniques.

3.2 Bucket Sorted Independent Sets Algorithm

The BSIS algorithm creates a bucket data structure making it possible to search for new C-points without individually scanning each vertex and its edges. The number of buckets in the data structure is $\max_{i \in S} |S_i^T|$ times the number of colors in the graph. Each possible weight in S has its own bucket. The vertices are dropped into the appropriate buckets during the setup of the coarse grid selection algorithm. To build the coarse grid, BSIS iterates by making vertices in the bucket with largest weight that is not empty into C-points (see Corollary 2) and removing them from the data structure. The weights of vertices affected by that operation are then updated. Some vertices will become F-points and are removed from the data structure. The affected vertices that do not become F-points are moved into lower weight buckets. These operations are repeated until all buckets are empty, at which point the coarse grid selection is complete. The following pseudo-code outlines the operations discussed above.

- Data structure setup:
  1: **for all** $i \in V$ **do**
  2: $bucketID \leftarrow (w_i - 1) \cdot numColors + color_i$
  3: $bucket[bucketID]$.INSERT(i)
- Independent set selection (search):
  1: **return** non-empty *bucket* with largest $bucketID$
- Weight update for vertex i:
  1: $bucketID \leftarrow (w_i - 1) \cdot numColors + color_i$
  2: $bucket[bucketID]$.REMOVE(i)
  3: $bucket[bucketID - numColors]$.INSERT($i$)

Figure 2 illustrates the bucket data structure. Not all buckets are represented in the figure. The vertices whose buckets are not shown are labeled with a gray number instead of a white number. The bucket a vertex is placed in depends on the number of vertices it strongly influences and its color. For example, vertex 2 strongly influences seven vertices and is black. Therefore, it was placed into the bucket shown furthest right in the figure. Notice the vertices in a bucket form an independent set (e.g., vertices 4, 13, 16, and 19).

The weight update routine described in Section 2.3 is very expensive in this context because in some iterations BSIS selects very few C-points. For a graph with a large number of colors, it may take BSIS dozens or hundreds of low-cost iterations to select a coarse grid. Recall the weight update routine described loops through all unassigned vertices each time it is called, so when it is run by BSIS, work is done on many unaffected vertices, which wastes computational time.

The largest factor in weight update cost results from searching for the conditions in Figure 1(b), which is done by looping through all unassigned vertices because in a CSR matrix, a new C-point cannot easily determine which vertices it strongly influences. It is much cheaper in this situation to construct the transpose of S than to search the entire graph in each iteration. In S^T, a C-point can quickly determine which vertices it influences and "paint them". The algorithm then loops through the painted vertices and determines if any are neighbors. This is a simple solution that has a dramatic effect on the performance of BSIS, although the update cost remains approximately equivalent to the cost in CLJP-c. Section 5 describes plans for continued investigation into decreasing the cost of weight update in coarsening algorithms.

4 Experimental Results

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

$$\begin{aligned} -\Delta u &= 0 \quad \text{on } \Omega \qquad (\Omega = (0,1)^3), \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{4}$$

This problem was chosen because it is a common initial test problem and structured problems often lead to the largest operator complexities for these algorithms. By creating larger operator complexities, the algorithms are forced to traverse more edges in the graph, leading to more work.

The algorithms were run on a single processor on Thunder, a large parallel machine at Lawrence Livermore National Laboratory. The 8GB of memory per node on Thunder allowed very large single-processor problems to be run. Timing data for the setup, search, and update portions of the coarse grid selection algorithms is reported. Timings such as these do not typically have a significant impact on the results, but due to the low-cost of the search in the BSIS, the timings do have a noticeable effect. The insight provided, however, motivates reporting the split timings. AMG solve phase information is not reported since the algorithms produce identical coarse grids, and information on solve phase performance for AMG with CLJP-c is documented in [1, 2].

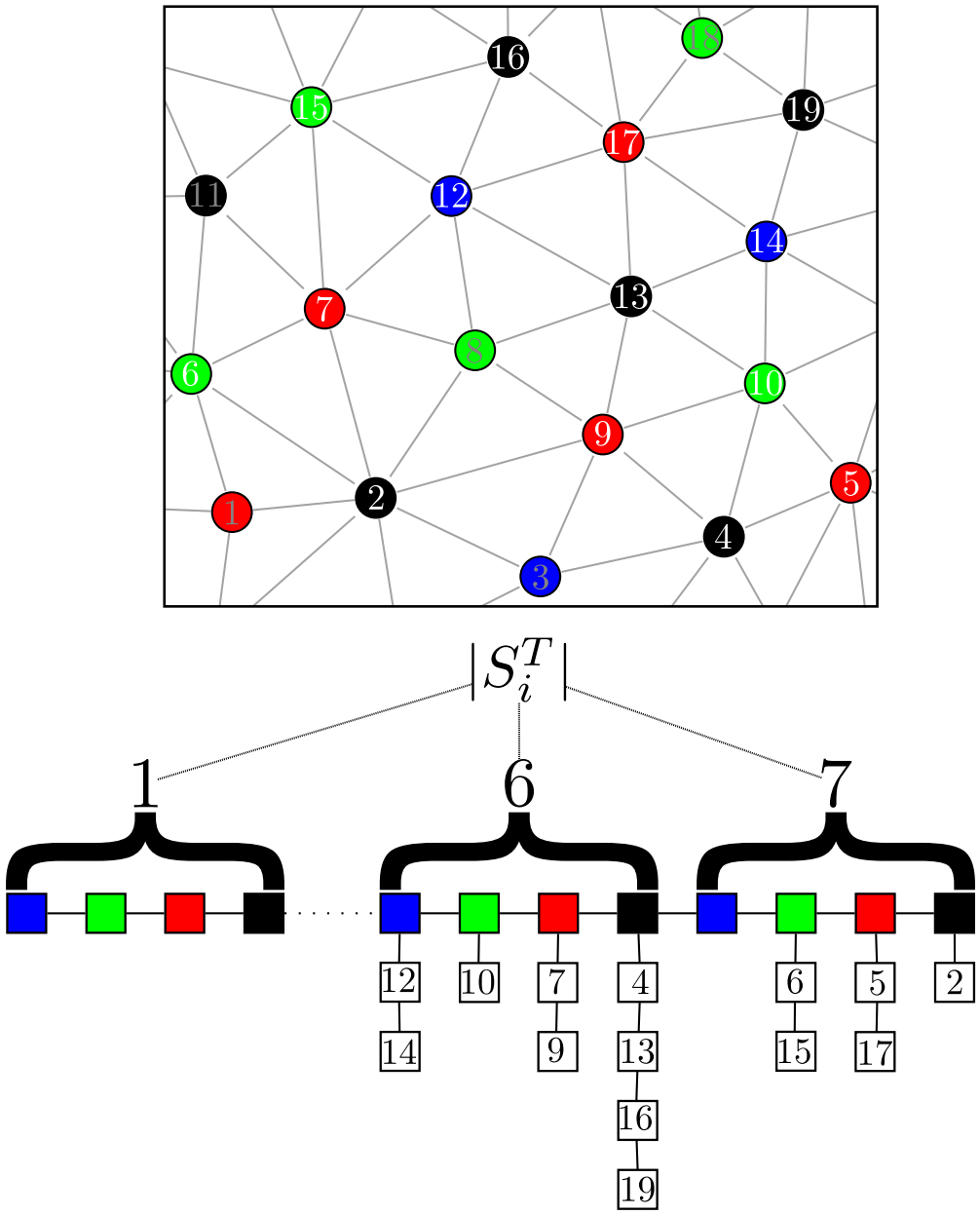


Figure 2: The BSIS data structure. Vertices in the mesh whose buckets are not represented are labeled with gray numbers. Undirected edges are shown, which implies strong connections in both directions.

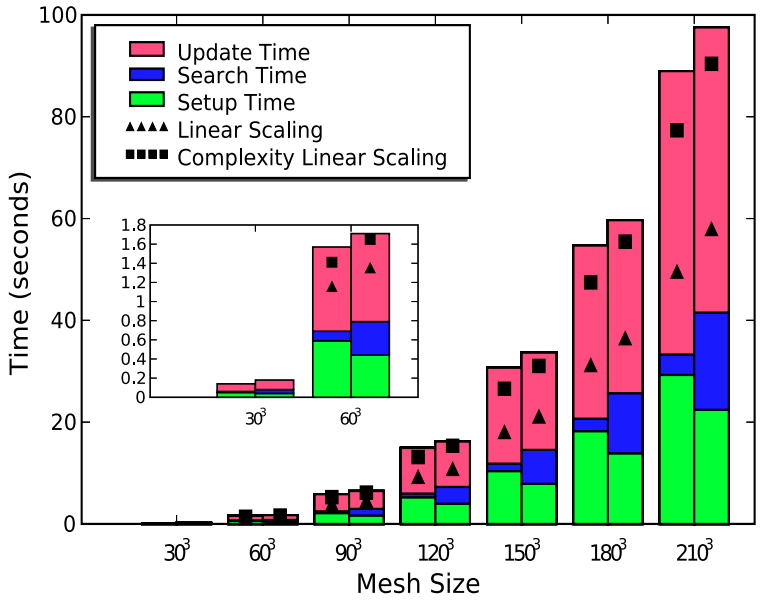


Figure 3: Coarse grid selection times using BSIS (left bars) and CLJP-c (right bars) on the 7-point Laplacian. The triangular dots represent linear scaling from the smallest problem to larger sizes relative to growth in the number of vertices, and the square dots represent linear scaling relative to the growth in the number of edges.

The smallest problem is on a $30 \times 30 \times 30$ grid. Subsequent problems are on grids of size 60^3, 90^3, and so on, up to 210^3. The largest problem is 343 times larger than the smallest problem and contains more than nine million degrees of freedom (i.e., vertices in the graph of S).

Results for the experiment are presented in Figure 3. BSIS completed coarse grid construction in less time than CLJP-c in every case. Search time was improved, and is likely significantly better than recorded without the profiling code active. It is important to note that although search was improved, the setup time cost increased. This is to be expected since more work is being done to initialize the data structure, but further work will be done to seek improvements. The triangles in the figure illustrate the linear scaling in time from the smallest problem. For example, if CLJP-c scaled linearly in cost with regards to the number of vertices, it would take less than sixty seconds to build the coarse grid for the largest problem. The squares are the linear scaling when operator complexity (a relative measure of the number of edges in S) is taken into account.

This experiment demonstrates the effectiveness and competitiveness of this method. BSIS is in an early stage of development, and we anticipate future work on this algorithm will yield increased efficiency and produce methods that are applicable to other coarsening algorithms and possibly to other elements in the AMG setup phase.

5 Conclusions and Future Work

This paper has introduced a new coarse grid selection algorithm called Bucket Sorted Independent Sets. BSIS features a search algorithm that does not traverse the graph to search for new C-points. This type of search is fast, and works independently of the edges in the graph. BSIS is designed to produce coarse grids identical to those built by CLJP-c more quickly than CLJP-c. This is demonstrated by the experiment in Section 4.

This work is active and further development on new combinatorial algorithms for coarse grid selection is planned. Future work is expected to improve the performance of this algorithm. Possible directions include developing methods to lower the cost of updating the bucket data structure, parallelization of BSIS, and more investigation into algorithms for lower cost vertex weight updates. The future work will focus on adapting more graph theory and graph algorithms to AMG.

References

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

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

[3] S. Blackford. Compressed row storage. Website: `http://www.cs.utk.edu/~dongarra/etemplates/node373.html`.

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

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

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

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

[8] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Compututing*, 15(4):1036–1055, 1986.

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

[10] H. De Sterck, U. M. Yang, and J. J. Heys. Reducing complexity in parallel algebraic multigrid preconditioners. *SIAM Journal on Matrix Analysis and Applications*, 27:1019–1039, 2006.

[11] K. Stüben. An introduction to algebraic multigrid. In *Multigrid*, pages 413–532. Academic Press, 2000.