Electronic Transactions on Numerical Analysis.
Volume 37, pp. 367-385, 2010.

ISSN 1068-9613.

ETNA
Kent State University
http://etna.math.kent.edu

COARSENING INVARIANCE AND BUCKET-SORTED INDEPENDENT SETS FOR ALGEBRAIC MULTIGRID*

DAVID M. ALBER† AND LUKE N. OLSON‡

Abstract. Independent set-based coarse-grid selection algorithms for algebraic multigrid are defined by their policies for weight initialization, independent set selection, and weight update. In this paper, we develop theory demonstrating that algorithms employing the same policies produce identical coarse grids, regardless of the implementation. The coarse-grid invariance motivates a new coarse-grid selection algorithm, called Bucket-Sorted Independent Sets (BSIS), that is more efficient than an existing algorithm (CLJP-c) using the same policies. Experimental results highlighting the efficiency of two versions of the new algorithm are presented, followed by a discussion of BSIS in a parallel setting.

Key words. Algebraic multigrid, parallel, coarse-grid selection.

AMS subject classifications. 65Y05, 65Y20, 65F10.

1. Introduction. The algebraic multigrid (AMG) method [5, 24] is an efficient numerical algorithm to iteratively approximate the solution to linear systems of the form $Ax = b$. Often, these algebraic systems arise from the discretization of partial differential equations on structured and unstructured meshes. In many cases, the computational complexity of AMG is $O(n)$, where n is the number of unknowns in the linear system. The linear cost property of AMG makes the method particularly attractive for large-scale problems.

Classical AMG algorithms execute in two phases: the setup phase and the solve phase. Setup phase algorithms provide functions for constructing elements needed by the solve phase, such as coarse grids, intergrid transfer operators (i.e., restriction and prolongation operators), and the linear systems to be solved on each level. The solve phase applies inexpensive iterative methods, called relaxation methods in the context of AMG, on the problems generated by the setup phase and uses the transfer operators to move vectors between levels.

In this paper, we focus attention on coarse-grid selection algorithms in the AMG setup phase and target the class of independent set-based coarsening algorithms in particular. New theory is developed to demonstrate invariance in the result of independent set-based coarse-grid selection, given restrictions on the selection policies. Algorithms that fall within the assumptions of the theory are guaranteed to produce the same coarse grids, regardless of how the policies are implemented. A new coarse-grid selection algorithm, designed to select coarse grids more efficiently than the CLJP-c algorithm, is presented to demonstrate the theoretical results. A theoretical benefit is that output from coarse-grid selection may be considered independent of the computational efficiency.

This paper is organized as follows. In Section 2, we discuss contemporary coarse-grid selection algorithms, and present the components of independent set-based coarsening algorithms. Section 3 develops theory based on general independent set-based coarsening, and the invariance of coarse grids selected when the same selection policies are used. Section 4 demonstrates the results from Section 3 through the introduction of a new coarsening algorithm called Bucket-Sorted Independent Sets (BSIS). BSIS produces the same coarse grids as the CLJP-c coarsening algorithm, but is more efficient by design. Experimental results

* Received June 10, 2010. Accepted September 9, 2010. Published online December 12, 2010. Recommended by Thomas A. Manteuffel.

†Microsoft Corp., One Microsoft Way, Redmond, WA 98052, U.S.A. (alber@engineering.uiuc.edu). This research was conducted while affiliated with the University of Illinois at Urbana-Champaign.

‡Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, U.S.A. (lukeo@illinois.edu).

ETNA
Kent State University
http://etna.math.kent.edu

are reported to demonstrate the performance gains in BSIS, and the integration of the BSIS algorithm into a parallel environment is discussed. Conclusions are presented in Section 5.

2. Coarse-grid selection. Coarse-grid selection algorithms construct the degrees of freedom for the coarse-level problems in AMG. Classical AMG implements nodal coarsening in which coarse-level degrees of freedom are a subset of the degrees of freedom on finer levels. Smoothed aggregation [9, 27, 28], on the other hand, forms a coarse-level degree of freedom by aggregating several fine-level degrees of freedom. In this paper, we study nodal coarsening.

Given a set of fine-level degrees of freedom, Ω_k, nodal coarse-grid selection constructs a set of coarse-level degrees of freedom $\Omega_{k+1} \subset \Omega_k$ based on the graph of matrix A. We refer to a CF-splitting of Ω_k as a separation of Ω_k into C-points, Ω_{k+1}, and F-points, $\Omega_k \setminus \Omega_{k+1}$. Several approaches have been developed to select the coarse grid, with independent set-based approaches being the most numerous [2]. Independent set-based algorithms descend from the classical AMG coarsening algorithm, i.e., Ruge-Stüben (RS) coarsening [24], also called classical coarsening. Independent set-based methods are the focus of this paper and are discussed in more detail in the next section.

Other methods take substantially different approaches to coarsening. The technique in [18] develops a parallel coarsening algorithm that utilizes RS. On each processor, RS selects several different coarse grids, and then one coarse grid is selected per processor in a way to minimize special treatment on processor boundaries. In [22], a greedy approach is used to produce a good splitting of the unknowns into fine-grid and coarse-grid sets. Subdomain blocking techniques [20] offer another approach for parallel coarse-grid selection by decoupling coarse grids and alleviating the need for communication on coarse levels. Compatible relaxation approaches [4, 6, 8, 21] form coarse grids incrementally and determine when the coarse grid is sufficient through a quality measure based on relaxation.

2.1. Independent set-based selection. Independent set-based coarsening algorithms rely on a *strength of connection* measure to estimate the correlation between degrees of freedom. In the classical definition, the set of unknowns that the ith unknown strongly depends on is based on the relative sizes of off-diagonal nonzero entries in the ith row of A, and is defined as

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

where $0 < \theta \leq 1$. The set of unknowns that i strongly influences, denoted S_i^T, is defined as the set of unknowns that strongly depend on i: $S_i^T = \{j : i \in S_j\}$. The strong influence and strong dependence sets are used in the definition of the *strength matrix* S, where the entries of S are defined as

$$S_{ij} = \begin{cases} 1, & \text{if } j \in S_i, \\ 0, & \text{otherwise.} \end{cases}$$

The classical strength definition is based on M-matrix properties and does not model the influence between unknowns in general cases. The formulation of a new, effective, and inexpensive strength of connection measures is an active area of research [7, 23], and has the potential to extend the applicability of AMG solvers.

Many independent set-based coarsening algorithms rely on strong influence and strong dependence to form the set of coarse-grid points Ω_{k+1}, using the following heuristics; see Figure 2.1.

ETNA
Kent State University
http://etna.math.kent.edu

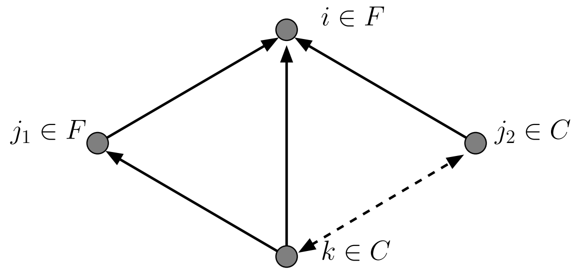

Figure 2.1: Demonstration of heuristic H1. Arrows indicate the direction of strong influence. Here, $i \in F$ is strongly influenced by j_1, j_2, and k. Both j_2 and k satisfy H1 by being C-points, while j_1 is an F-point satisfying H1 by having a common strongly influencing neighbor (k) with i.

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

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

In general, H1 and H2 cannot simultaneously be satisfied, so H1 is enforced by independent set-based methods, while H2 serves as a guideline whose purpose is to encourage selection of small coarse-grid point sets, i.e., small $|\Omega_{k+1}|$.

One artifact of H1 and H2 is the tendency to generate denser coarse operators, resulting in large operator complexities. As we discuss in Section 4, the sparsity of the graphs impacts the performance of coarsening algorithms and is a consideration when designing more efficient data structures. On the other hand, certain independent set-based coarsening algorithms use a variant of H1, called H1$'$,

H1$'$: each F-point must be strongly influenced by at least one C-point,

which is designed to produce sparser coarse-level matrices [14]. The theory and algorithms we develop in this paper target H1 as a guide, but they are also suitable for H1$'$-type coarsening approaches, such as color-based methods, that satisfy this heuristic.

Independent set-based coarsening algorithms operate on the graph of matrix S. Let $G(S) = (V, E)$ be the graph of S where V defines the vertex set consisting of the degrees of freedom and E is the set of edges in $G(S)$, where $(i, j) \in E$ if $S_{ij} = 1$.

Independent set-based coarse-grid selection algorithms implement three routines to select coarse grids: initialization, independent set selection, and weight update. A general independent set-based coarsening algorithm is outlined in Algorithm 2.1. Each vertex in the graph of S is assigned a weight based on its merit in becoming a C-point. The largest weights are associated with vertices best suited to be C-points, where the quality of potential C-points is quantified by strength of connection measures.

In classical coarsening, the independent set D in line 6 of Algorithm 2.1 contains a single vertex i in each iteration, where $w_i \geq w_j, \forall j \in V \backslash (C \cup F)$. Coarsening with independent sets containing a single vertex per iteration is not scalable, so parallel independent set methods use different weight initialization policies designed to enable the selection of larger independent sets. In this case D is selected by comparing weights in each node neighborhood.

DEFINITION 2.2. *The neighborhood $\mathcal{N}_i$ of vertex i, is the union of the strong influences of i and the strong dependencies of i: $\mathcal{N}_i = S_i \cup S_i^T$.*

For these parallel algorithms, vertex i is eligible to be added to the independent set D if

ETNA
Kent State University
http://etna.math.kent.edu

ALGORITHM 2.1: Independent Set-Based Coarse-Grid Selection

1 INDEPENDENT-SET-COARSE-GRID-SELECTION(S):
2 $F \leftarrow \emptyset$
3 $C \leftarrow \emptyset$
4 $w \leftarrow$ INITIALIZE-WEIGHTS(S)
5 **while** $C \cup F \neq V$
6     $D \leftarrow$ SELECT-INDEPENDENT-SET(w, S)
7     $C \leftarrow C \cup D$
8     $(F, w) \leftarrow$ UPDATE-WEIGHTS(D, F, S, w)
9 **return** C, F

---

the following condition is satisfied:

$$w_i > w_j, \quad \text{for all } j \in \mathcal{N}_i. \tag{2.1}$$

The weight update routines in independent set-based coarsening algorithms (line 8, Algorithm 2.1) affect weights of a subset of the vertices in $\bigcup_{i \in D} \mathcal{N}_i$ and take one of two forms: routines that decrease the values of weights and routines that increase the value of weights. In all cases, a vertex with a large weight has high fitness and is more desirable as a C-point. The pseudocode, as written in Algorithm 2.1, performs assignment of vertices to the F-point set during the weight update routine.

Many independent set-based coarsening algorithms have been developed. RS coarsening is introduced in [24] and is the inspiration for many algorithms that follow. The first parallel independent set-based coarsening algorithms appeared a little more than ten years ago and include RS3 [19], CLJP [12], BC-RS and Falgout [19], PMIS and HMIS [14], CLJP-c [1], PMIS-c1 and PMIS-c2 [2], and PMIS Greedy [10], to name a few. Independent set-based coarsening algorithms are defined by a set of *policies* that specify how weights are initialized, the neighborhood used in selecting C-points (see selection neighborhood below), and the way in which vertex weights are updated. We investigate the nature of these components in the following section.

3. Invariance of selection in independent set-based methods. In this section, we develop a theory to demonstrate that, under appropriate conditions, general independent set-based algorithms produce identical coarse grids. The motivation for developing this theory is to provide a more formal framework for independent set-based coarsening algorithms. The main advantages of providing this theory is to more clearly decompose the coarsening algorithm design and implementation choices from coarsening quality (see Section 4) and to provide insight into possible algorithms that have not been explored. This theory applies to many existing, well-known independent set-based coarsening algorithms, including CLJP, CLJP-c, PMIS, and HMIS.

An artifact of the theory presented in this section is that algorithms relying on different and larger neighborhoods, such as *distance-d neighborhoods*, may be used to develop additional coarsening algorithms.

DEFINITION 3.1. *The distance-d neighborhood $\mathcal{N}_i^d$ of vertex i is the set of vertices j that are distance d or less from i in the symmetrized strength graph, excluding i. That is,*

$$\mathcal{N}_i^d = \left\{ \bigcup_{j \in \mathcal{N}_i^{d-1}} (\mathcal{N}_j \cup \{j\}) \cup \mathcal{N}_i \right\} \setminus \{i\},$$

ETNA
Kent State University
http://etna.math.kent.edu

where $d \geq 0$*,* $\mathcal{N}_i^0 = \emptyset$*, and* $\mathcal{N}_i^1 = \mathcal{N}_i$*.*

While the distance-d neighborhood, $\mathcal{N}_i^d$, is a natural definition for the set of points from which the C-points are selected, we do not limit our theory to this choice. We introduce a more abstract set, called the *selection neighborhood*, which is defined in the following

DEFINITION 3.2. *The selection neighborhood* $\mathcal{N}_i^s$ *is the set of vertices for vertex* i *from which potential* C*-points are selected.*

Furthermore, given a selection neighborhood, we define the *general selection criterion* through the following

DEFINITION 3.3. *An independent set-based coarse-grid selection algorithm satisfies the general selection criterion if a given vertex* i *is eligible to be added to* D *whenever*

$$w_i > w_j, \quad \text{for all } j \in \mathcal{N}_i^s, \tag{3.1}$$

where $\mathcal{N}_i^s$ *is the selection neighborhood of* i*.*

For example, $\mathcal{N}_i^s = \mathcal{N}_i^1$, $\mathcal{N}_i^s \subset \mathcal{N}_i^1$, or $\mathcal{N}_i^s = \mathcal{N}_i^2$, are common choices for the selection neighborhood. It is assumed, however, that the matrix formed by $\mathcal{N}_*^s$ (i.e., the selection sets for all vertices) is symmetric. This is equivalent to stating if $j \in \mathcal{N}_i^s$, then $i \in \mathcal{N}_j^s$ for all i and j in the vertex set.

We proceed with the following additional assumptions. Vertex weights are assigned such that larger values signify greater value for inclusion on the coarse grid. The weight of vertex i never increases, the weight decreases only as a result of weight updates, and the weight is unaffected by the assignment of a vertex $j \notin \mathcal{N}_i^s$ to the C-point set. Furthermore, weight updates are deterministic, in the sense that an arbitrary set of new C-points results in the same weights after update, regardless of the order in which the C-points are processed. Finally, algorithms to date decrement weights by one for each edge that is symbolically removed from the strength graph, but the theory below applies more broadly, allowing algorithms that decrease weights by values other than one following the removal of an edge. Furthermore, it is assumed that there is a consistent approach to breaking ties in the case of equal weights; this results in a deterministic and terminating selection algorithm.

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

DEFINITION 3.4. *The extended selection neighborhood* $\mathcal{N}_i^{2s}$ *of vertex* i*, is the union of the selection neighborhood* $\mathcal{N}_i^s$ *with the selection neighborhoods of the vertices in* $\mathcal{N}_i^s$*, excluding* i*. That is,*

$$\mathcal{N}_i^{2s} = \left\{ \left(\bigcup_{j \in \mathcal{N}_i^s} \mathcal{N}_j^s \right) \cup \mathcal{N}_i^s \right\} \setminus \{i\}.$$

When a vertex i satisfies the generalized selection criterion (3.1), no other valid C-point assignments are able to affect w_i. This is formalized below.

LEMMA 3.5. *If (3.1) is satisfied for vertex* i*, then* i *is the next vertex in* $\{i\} \cup \mathcal{N}_i^s$ *to become a* C*-point, regardless of other* C*-point selections and corresponding weight updates made in the graph.*

Proof. Assume (3.1) holds; that is, $w_i > w_j$ for all $j \in \mathcal{N}_i^s$. Suppose vertex $k \neq i$ is next selected as a C-point.

Case 1: If $k \in \mathcal{N}_i^s$, then the assumption is violated because $w_i \not> w_k$.

Case 2: If $k \notin \mathcal{N}_i^s$, then weights w_j, for $j \in \mathcal{N}_k^s$, are decremented. Since the selection neighborhood is symmetric, $i \notin \mathcal{N}_k^s$ so that (3.1) is still maintained for vertex i. Thus $w_i > w_j, \forall j \in \mathcal{N}_i^s$, holds until vertex i is selected as a C-point. $\square$

ETNA
Kent State University
http://etna.math.kent.edu

Let D_0 be the set of all vertices satisfying (3.1) following weight initialization. Vertices in D_0 become C-points regardless of the algorithm used to build coarse grids. In most independent set-based algorithms, all D_0 vertices become C-points in the first iteration. Any algorithm constructed, however, eventually selects all D_0 vertices as C-points, as proven in the following corollary.

COROLLARY 3.6. *Consider a CF-splitting $\Omega_k = F \uplus C$, based on (3.1). A given selection neighborhood $\mathcal{N}_*^s$ and weight update method results in $D_0 \subset C$.*

Proof. The proof follows from Lemma 3.5. For each $i \in D_0$, (3.1) is satisfied, thus each $i \in D_0$ is assigned to C without effect on other $j \in D_0$. ∎

Corollary 3.6 states that any coarse-grid selection method using the general selection condition invariably selects D_0 vertices as C-points. Given the conditions developed earlier in this section, we now prove that coarse-grid selection is insensitive to the implementation decisions made in specific algorithms.

THEOREM 3.7. *Let w be a set of initial weights and $\mathcal{N}_*^s$ be a symmetric selection neighborhood. All independent set-based coarse-grid selection algorithms satisfying the general selection criterion (see Definition 3.3) and implementing the same weight reduction policy yield identical coarse grids.*

Proof. Consider two vertices $i \in \mathcal{N}_j^s$ and $j \in \mathcal{N}_i^s$, and let CGS$_A$ be a coarse-grid selection algorithm satisfying the assumptions of the theorem. Furthermore, assume that, in CGS$_A$, vertex i is assigned to C before j, i.e., either j is assigned to C after i or j is assigned to F at any time. To prove the theorem, we show that there is no coarse-grid selection algorithm CGS$_B$ that is similar to CGS$_A$ as assumed in the theorem for which j is assigned to C before i.

Let CGS$_A$ and CGS$_B$ make identical C-point selections up to an arbitrary iteration before i or j are added to C. At this iteration, let D_i^{2s} be the set of all nodes in the extended selection neighborhood of i satisfying (3.1). Without a loss of generality, assume CGS$_A$ next selects subset $A \subset D_i^{2s}$ (i.e., $C \leftarrow C \cup A$) by which $w_i > w_k$ for all $k \in \mathcal{N}_i^s$, and assume CGS$_B$ selects subset $B \subset D_i^{2s}$ (i.e., $C \leftarrow C \cup B$) by which $w_j > w_k$ for all $k \in \mathcal{N}_j^s$. Therefore, in CGS$_A$, i becomes a C-point before j, and in CGS$_B$, j becomes a C-point before i.

For B to exist, at least one of the following necessary conditions must hold:

Case 1: the value of w_i is smaller when B is added to the C-point set than when A is added, that is, $w_i : C \leftarrow C \cup B$ is less than $w_i : C \leftarrow C \cup A$;

Case 2: the value of w_j is larger when B is added to the C-point set than when A is added, that is, $w_j : C \leftarrow C \cup B$ is greater than $w_j : C \leftarrow C \cup A$.

Case 1: this condition implies the existence of $\ell \in B \cap \mathcal{N}_i^s$, $\ell \notin A$. Thus, after $C \leftarrow C \cup A$, $w_\ell > w_i$, contradicting the definition of A.

Case 2: this condition implies the existence of $m \in A \cap \mathcal{N}_j^s$, $m \notin B$. Thus, after $C \leftarrow C \cup B$, $w_m > w_j$, contradicting the definition of B.

Since both conditions result in a contradiction, we conclude that A and B cannot both exist, and therefore, j cannot be selected as a C-point before i. ∎

Theorem 3.7 is an important result about the nature of coarse grids selected by general independent set-based algorithms. This information enables the design and implementation of new algorithms that yield identical coarse grids using different and possibly more efficient techniques.

4. Bucket-Sorted Independent Sets selection. In this section, we introduce the Bucket-Sorted Independent Set (BSIS) algorithm. BSIS uses the same policies as the CLJP-c algorithm [1]. Following the theory in Section 3, BSIS and CLJP-c produce the same coarse grids. BSIS, however, is designed to select independent sets more efficiently than CLJP-c.

ALGORITHM 4.1: CLJP-c Weight Initialization

```
1 CLJP-C-INITIALIZE-WEIGHTS(S):
2 σ ←COLOR(G(S))              /* returns vertex−to−color mapping */
3 c_ℓ = 1 . . . max σ          /* colors numbered [1, # of colors] */
4 for  c ∈ c_ℓ
5     c_w(c) ← (c − 1)/|c_ℓ|
6 for  i ∈ V
7     w(i) ← |S_i^T| + c_w(σ(i))
8 return  w
```

4.1. CLJP-c. Before selecting C-points, CLJP-c colors the graph of S such that each pair of adjacent vertices in S are of different color. Colors are used as one component of vertex weights, and the number of strong influences provides the other source of contribution. Initialization of the weights in CLJP-c is detailed in Algorithm 4.1. As a result of coloring, the structure of the coarse grids selected is improved over the CLJP algorithm [12]. CLJP-c was initially introduced with a random perturbation term in line 7 of Algorithm 4.1. The perturbation was designed to serve the same purpose as perturbations in CLJP, namely to break ties in vertex weights, but is not included here because its purpose is fulfilled entirely by graph coloring as shown in the following theorem.

THEOREM 4.2. *For any i and for each $j \in \mathcal{N}_i$, CLJP-c guarantees $w_i \neq w_j$.*

Proof. Consider any node i and assume there exists $j \in \mathcal{N}_i$ such that $w_i = w_j$. Then, from Algorithm 4.1 we have

$$|S_i^T| + c_w(\sigma(i)) = |S_j^T| + c_w(\sigma(j)).$$

Consequently, $|S_i^T| = |S_j^T|$ and, without loss of generalization, $c_w(\sigma(j)) = c_w(\sigma(i))$. Since the color weights are separated by at least

$$\frac{1}{|c_\ell|} = \inf\{|c_w(k_1) - c_w(k_2)| : c_w(k_1) \neq c_w(k_2)\},$$

we have that $\sigma(j) = \sigma(i)$. However, S is colored such that $\sigma(i) \neq \sigma(j)$ for neighboring i and j ($j \in \mathcal{N}_i$), resulting in a contradiction. ∎

Theorem 4.2 establishes that all neighbor vertices have different weights in CLJP-c, which, although unlikely to occur, is not guaranteed in CLJP. The following corollaries are a result of Theorem 4.2.

COROLLARY 4.3. *Any set of vertices in the graph of S sharing the same weight is an independent set.*

COROLLARY 4.4. *The set of vertices sharing the largest weight in the graph of S forms an independent set satisfying (3.1).*

Corollary 4.3 states that independent sets can be selected in the graph simply by selecting sets of vertices with the same weight. Corollary 4.4 refines this observation to a subset of vertices guaranteed to satisfy the selection criterion (3.1). In particular, it shows it is possible to build the coarse grid by selecting vertices with the maximum weight in the graph, updating weights, selecting the next set of vertices with maximum weight, and so on. This is the approach taken by the BSIS algorithm.

CLJP and CLJP-c rely on a search routine to locate all vertices satisfying (3.1) (i.e., vertices with locally maximal weights) and on a weight update routine to modify weights of vertices in the neighborhoods of new C-points.

ETNA
Kent State University
http://etna.math.kent.edu

ALGORITHM 4.5: CLJP Independent Set Selection

```
1 CLJP-SELECT-INDEPENDENT-SET(S, C, F):
2 D ← ∅
3 for i ∈ V \ (C ∪ F)
4     if w_i > w_j for all j ∈ N_i
5         D ← D ∪ {i}
6 return D
```

The algorithms for searching and updating vertex weights in CLJP are examined in this section in detail. The pseudo-code below assumes the software uses a compressed sparse row (CSR) [25] matrix format or other similar format, which are common sparse matrix formats in numerical software. CSR provides low memory costs for storing sparse matrices and provides efficient access to the nonzeros in a row. Accessing the nonzeros in a column is an expensive operation in this format and strongly influences the weight update routine in CLJP-style algorithms because S is generally not symmetric.

The search step performed in CLJP independent set selection is outlined in Algorithm 4.5. The inner loop (encapsulated in line 4) is executed $2|E|$ times in the first call to Algorithm 4.5. The total search cost of constructing the coarse grid is complicated to analyze and depends on the number of iterations needed, in addition to other factors. In the best case of $\Omega(E)$ time, the cost is significant when the graph contains large numbers of edges, as usually happens on the lower levels in the grid hierarchy (see [2] for examples). In Section 4.2, we introduce the technique used in BSIS, which conducts the search independent of the number of edges in the graph.

Pseudo-code for the weight update in CLJP is shown in Algorithm 4.6. 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 (lines 3–8) is trivial, since determining the vertices in S_k is inexpensive. The second type of update (lines 10–23) is more expensive, since the vertices influenced by k are difficult to determine in a CSR format. The update requires a search of many vertices i and all of their strong influencing neighbors j. The routine then searches strongly influencing j to determine if any $k \in D$ strongly influences both i and j. The cost increases dramatically as the density of S increases. Large operator complexities have a disproportionately large impact on coarse-grid selection run time. In the next section, a modified update routine to complement the new search technique is introduced.

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

Like CLJP-c, BSIS depends on graph coloring, but utilizes modified routines for search and weight update. Furthermore, rather than applying the color information to augment vertex weights, BSIS uses the colors in a bucket data structure. Once initialized, this data structure selects independent sets, which satisfy the conditions in (2.1), in constant time.

Figure 4.1 illustrates the bucket data structure. The number of buckets in the data structure is $\max_{i\in V} |S_i^T|$ times the number of colors in the graph, that is, each possible weight in S has its own bucket. The vertices are distributed to the appropriate buckets during the setup of the coarse-grid selection algorithm, where the bucket of a newly placed vertex depends

ETNA
Kent State University
http://etna.math.kent.edu

ALGORITHM 4.6: CLJP Weight Update for CSR Matrix

```
UPDATE-WEIGHTS(D, C, F, S, w):
S̄ ← S
for d ∈ D
   for i ∈ S_d
      w_i ← w_i − 1
      S_d = S_d \ {i}                          /*removing edge from graph*/
      if w_i < 1
         F ← F ∪ {i}
for i ∈ V \ (C ∪ F)
   N_i ← ∅
   for k ∈ S̄_i ∩ D
      N_i ← N_i ∪ {k}                          /*mark k new C−point*/
   for j ∈ S_i
      if j ∉ D
         for k ∈ S_j ∩ N_i     /* i and j mutually influenced by k */
            w_j ← w_j − 1
            S_i = S_i \ {j}                    /*remove edge from j to i*/
            if w_j < 1
               F ← F ∪ {j}
   for k ∈ S_i ∩ D
      S_i = S_i \ {k}                          /*remove edge from k to i*/
return F, w
```

ALGORITHM 4.7: BSIS Data Structure Setup

```
BSIS-SETUP(S, w, σ):
B ← BSIS-ALLOCATE-BUCKETS(max w_i, max σ)
for i ∈ V
   b ← (w_i − 1) · max σ + σ(i)
   B[b].INSERT(i)
```

on the number of vertices it strongly influences and its color. For example, in Figure 4.1 vertex 14 strongly influences six vertices and is black. Therefore, it is placed into the black bucket in the sixth group of buckets. Notably, the vertices in a bucket form an independent set; e.g., vertices 14, 16, and 21.

In each iteration, the non-empty bucket with largest weight forms D; see Corollary 4.4. These vertices are assigned to the C-point set and removed from the data structure. Vertex weight updates lead to corresponding updates to the data structure, and new F-points are removed from the data structure. These operations continue until all buckets are empty, at which point the coarse-grid selection is complete. Algorithms 4.7, 4.8, and 4.9 outline the operations discussed above.

Figure 4.2 illustrates the graph and data structure following the first iteration of the algorithm. Vertex 10 has become a C-point and its neighbors weights have been updated. Vertices assigned to F or C (highlighted with a red ring in Figure 4.2) are removed from the data structure, and other affected vertices are moved to new locations in the data structure.

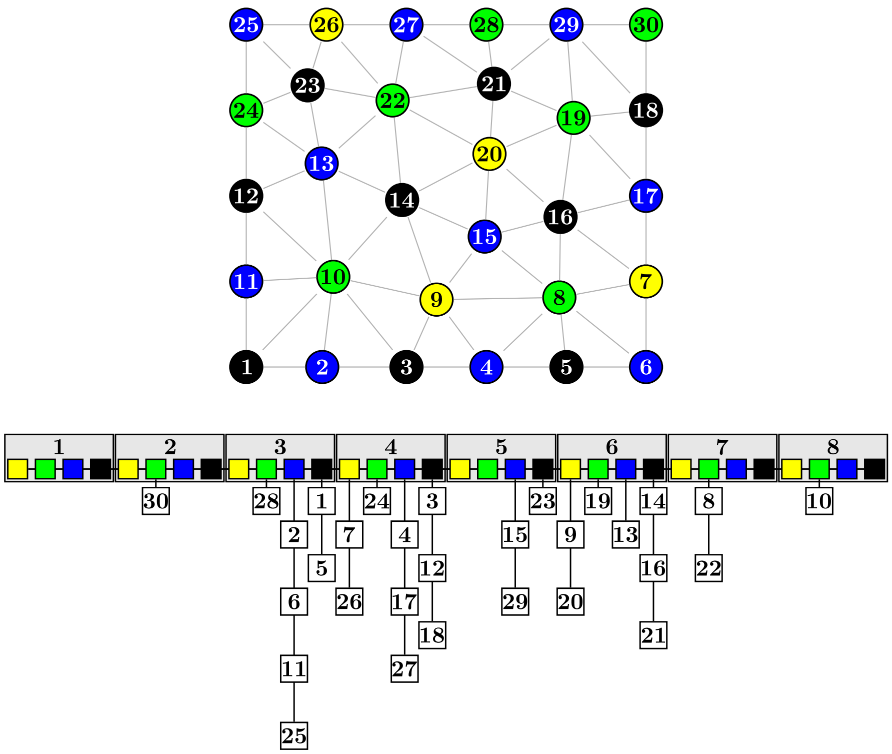

Figure 4.1: Graph of a strength matrix and its BSIS data structure. The edges in the graph represent mutual strong influence between vertices.

ALGORITHM 4.8: Independent Set Selection

```
BSIS-SELECT-INDEPENDENT-SET(B):
  return non−empty bucket with largest bucket ID
```

Using the weight update routine described in Algorithm 4.6 with BSIS is very expensive, because some iterations of BSIS select very few C-points. For a graph with a large number of colors, BSIS may execute dozens or hundreds of low-cost iterations to select a coarse grid, resulting in a large range of memory access. This is due to the weight update routine looping through all unassigned vertices at each call.

The largest factor in the cost of the weight update in Algorithm 4.6 results from searching for the second type of weight update (beginning at line 10), which must be done by looping through all unassigned vertices, because a new C-point is unable to easily determine which vertices it strongly influences in a CSR matrix. It is less expensive, in this situation, to construct the transpose of S than to search the entire graph in each iteration. In S^T, a C-point quickly determines which vertices it influences and "paints" them. The algorithm then loops through all painted vertices and determines if any are neighbors. This simple solution has a dramatic effect on the performance of BSIS, although the update cost remains approximately equivalent to the cost in CLJP-c.

ETNA
Kent State University
http://etna.math.kent.edu

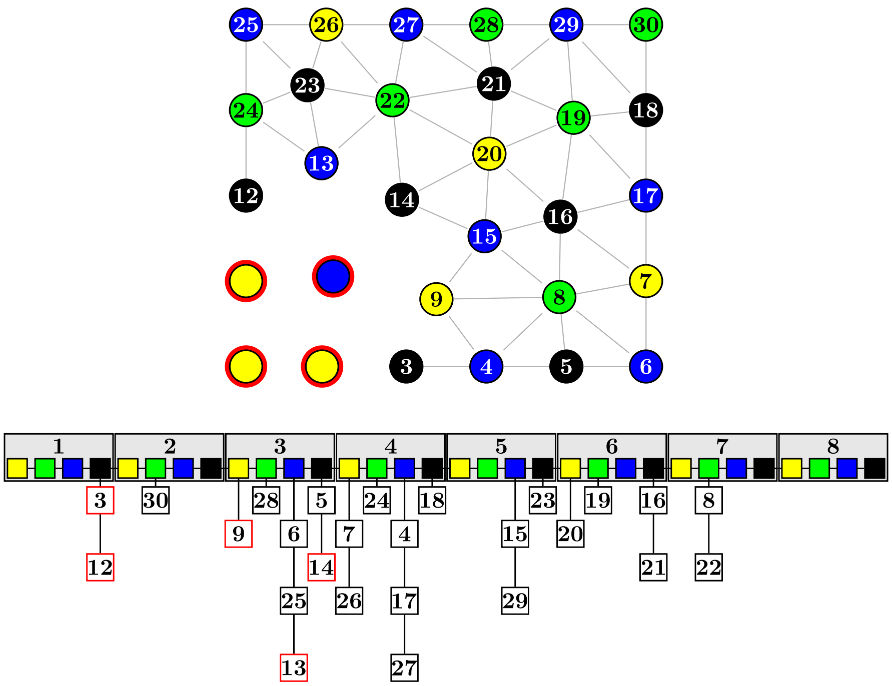

Figure 4.2: BSIS data structure following selection of the first C-point (vertex 10, now solid blue and unnumbered). The weights of neighbors of the new C-point are updated. Some neighbors become F-points (shown as unnumbered yellow vertices) and are removed from the data structure. Vertices removed from the data structure are highlighted with a red ring in the graph, while other neighbors are moved to new locations in the data structure and are highlighted (to aid in reading the figure) in the data structure with a red box.

ALGORITHM 4.9: BSIS Edge-Removal Weight Update

BSIS-UPDATE-WEIGHTS(B, σ, i):
$b \leftarrow (w_i - 1) \cdot \max \sigma + \sigma(i)$
$B[b]$.REMOVE(i)
$B[b - \max \sigma]$.INSERT(i)

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

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

To save cost, we suggest an approach that aggregates the cost of updating vertex weights. Rather than investing computation into maintaining accuracy in the data structure, a less

ETNA
Kent State University
http://etna.math.kent.edu

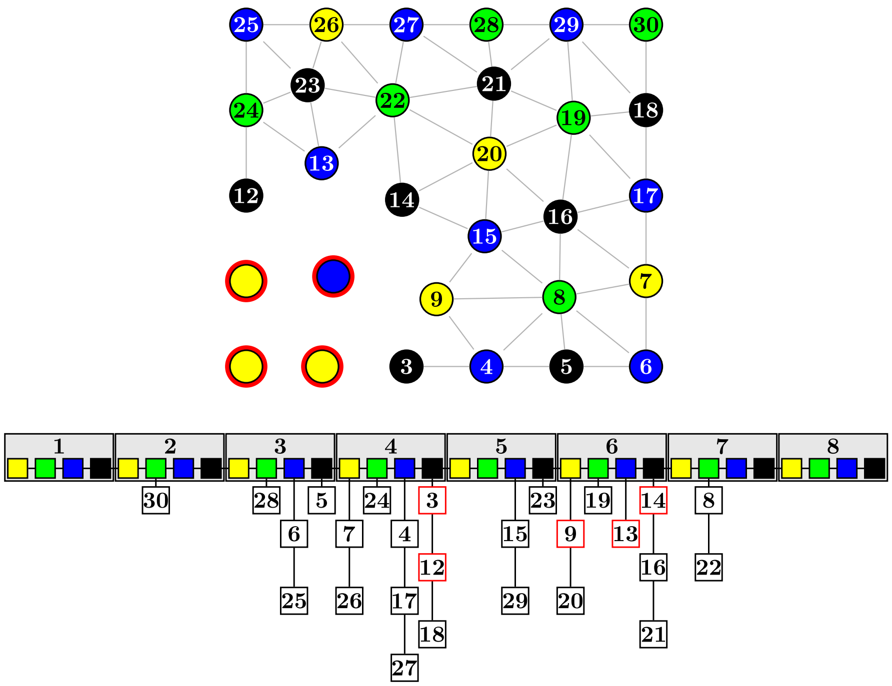

Figure 4.3: BSIS data structure after selecting the first C-point (vertex 10) with aggregate weight updates.

expensive mechanism to test if a vertex is in the correct location is provided. When a weight is updated, the vertex is not moved until it is found in the bucket being used as the new independent set D.

Figure 4.3 depicts the data structure after the first set of C-points is selected. Rather than moving vertices to new buckets, the method now keeps them in the same location, and only moves them when necessary. As shown in Section 4.4, aggregation of the weight updates leads to significant savings in computational cost.

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

$$-\Delta u = f, \quad \text{on } \Omega \quad (\Omega = (0,1)^3).$$

This problem, and the two to follow, have homogeneous Dirichlet boundary conditions.

BSIS is implemented as described in Sections 4.2 and 4.3. In particular, BSIS employs the transpose of the strength matrix, and the timings below include the time to compute this transpose. CLJP-c, on the other hand, works directly from the strength matrix. The strength factor θ is 0.25 in this and the following experiments.

The experiments were serial and run on a single processor of the Thunder supercomputer at Lawrence Livermore National Laboratory. Timing data for the selection of all coarse grids in the hierarchy is reported. This time includes the cost for all parts of the algorithm, including the graph coloring phase. 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].

ETNA
Kent State University
http://etna.math.kent.edu

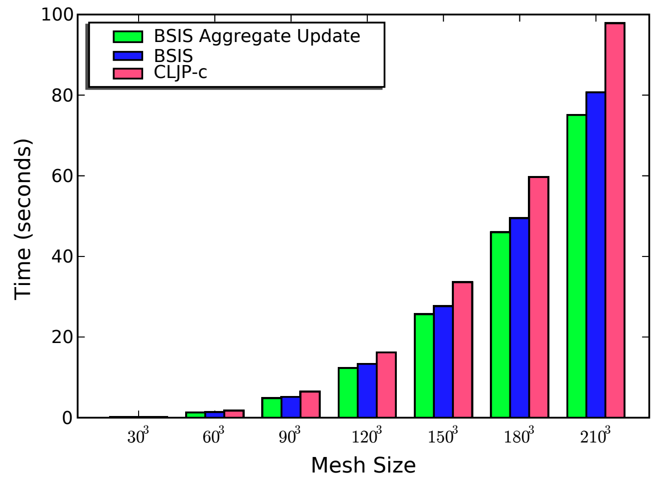

Figure 4.4: Coarse-grid selection times using BSIS with aggregate weight update, standard BSIS, and CLJP-c on the 7-point Laplacian.

The smallest problem is a $30 \times 30 \times 30$ grid. Subsequent problems are 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.

Results for the experiment are presented in Figure 4.4. BSIS completes coarse-grid construction in less time than CLJP-c in every case, and BSIS with aggregate weight update performs better than standard BSIS. For the largest problems BSIS is approximately 17% less expensive than CLJP-c, while BSIS with aggregate weight updates is 23% less expensive on the largest problems. The benefit is magnified, relative to CLJP-c, for the smaller problems.

For our second experiment, we compare the running time of implementations of BSIS, with aggregate weight updates, and CLJP-c, using the PyAMG [3] package. In this experiment, we provide CLJP-c with the transpose of the strength matrix, which makes the separation in timings between the methods substantially smaller.

The problem is a finite element discretization of a two-dimensional Laplacian problem on a Cartesian grid:

$$-\Delta u = f, \quad \text{on } \Omega \quad (\Omega = (0,1)^2). \tag{4.1}$$

Timings for this problem include all portions of the setup phase, except the cost of computing the transpose of the strength matrix, since both algorithms utilize it. Data points were produced 100 times each, and the mean values are reported.

Figure 4.5 plots timings and the timing gap between CLJP-c and BSIS with aggregate weight update, when both are provided the transpose of the strength matrix. The smaller difference in timings compared to those in the first test is due to at least two factors. First, this two-dimensional problem is less taxing for coarsening algorithms, because it produces a coarse grid hierarchy with fewer nonzeros; second, providing CLJP-c with the transpose of the strength matrix results in a performance boost for CLJP-c. The plot in Figure 4.6 shows that the time spent selecting independent sets for BSIS with aggregate weight updates is the source of its lower cost.

ETNA
Kent State University
http://etna.math.kent.edu

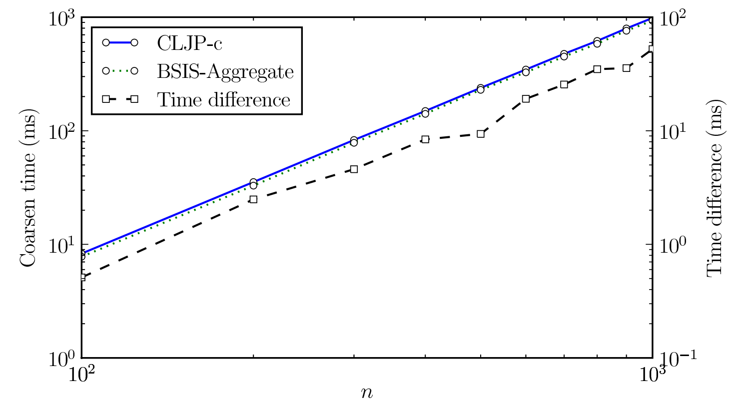

Figure 4.5: Coarse-grid selection times using BSIS with aggregate weight update and CLJP-c on the two-dimensional finite element Laplacian problem (4.1). The figure plots total time for coarsening on the left y-axis and the difference in times on right y-axis. The number of unknowns per trial is n^2.

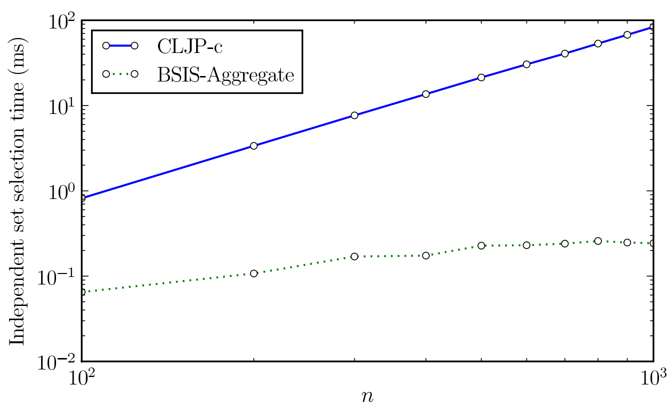

Figure 4.6: Total time for selecting independent sets in CLJP-c and BSIS with aggregate weight updates for the two-dimensional finite element Laplacian problem (4.1). The number of unknowns per trial is n^2.

CLJP-c spends about ninety milliseconds more than BSIS with aggregate updates, selecting independent sets on the largest problem. The total time difference between the methods on the largest problem is only about fifty milliseconds, however. The discrepancy between time saved by improved independent set selection and total time saved results from the weight update time, which is more efficient when larger independent sets are given. Since BSIS generates smaller independent sets and runs the weight update subroutine many more times than CLJP-c, there is a noticeable performance impact. Finally, note that while the time spent by CLJP-c computing the independent set consistently grows as n increases, BSIS with aggregate weight update begins to show total selection times that are independent with respect to n, for larger values of n.

ETNA
Kent State University
http://etna.math.kent.edu

The final problem in our experiments set is a rotated anisotropic diffusion equation on a Cartesian grid, discretized using finite elements:

$$-\nabla \cdot QAQ^T \nabla u = f, \tag{4.2}$$

where

$$Q = \begin{bmatrix} -0.5 & -\sqrt{3}/2 \\ \sqrt{3}/2 & -0.5 \end{bmatrix} \qquad \text{and} \qquad A = \begin{bmatrix} 1 & 0 \\ 0 & 0.0001 \end{bmatrix}.$$

As in the previous problem, timings include all portions of the setup phase work, except the cost of computing the transpose of the strength matrix. The timings reported are the means of 100 samples for each data point.

Due to the anisotropy, the graph of the matrix in (4.2) is coarsened more slowly than the graph of the matrix in (4.1), leading to a greater number of levels in the coarse-grid hierarchy and a larger number of calls to the weight update routine. Figure 4.7 plots the timing information for CLJP-c and BSIS with aggregate weight updates, which shows that the time difference is smaller for this problem. As plotted in Figure 4.8, the gap between independent selection for the two methods is slightly larger than in the finite element Laplacian problem (4.1), because the selection time for CLJP-c is higher. In spite of approximately the same savings in selection time, compared with the previous problem, the total savings is lower, due to an increase in the gap of weight update times of the two methods.

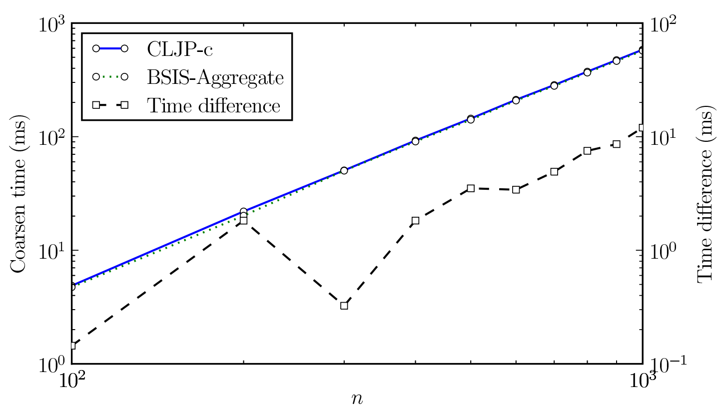

Figure 4.7: Coarse-grid selection times using BSIS with aggregate weight update and CLJP-c on the rotated anisotropic diffusion problem (4.2). The figure plots total time for coarsening on the left y-axis and the difference in times on right y-axis. The number of unknowns per trial is n^2.

The experiments demonstrate the effectiveness and competitiveness of the bucket technique. The independent set selection approach in BSIS is shown to be more efficient and scale better than the selection routine in CLJP-c. While BSIS with aggregate weight update is less expensive than CLJP-c in the experiments, the gains obtained by the improved selection routine can be eroded by excess cost in the weight update routine. We anticipate that further efficiency for weight update and the BSIS algorithm is possible, for example through hybrid techniques that result in slightly different, but less expensive coarse grid hierarchy construction. Furthermore, the methods and concepts in this research are also applicable to other coarsening algorithms, such as color-based methods designed to satisfy heuristic H1′.

ETNA
Kent State University
http://etna.math.kent.edu

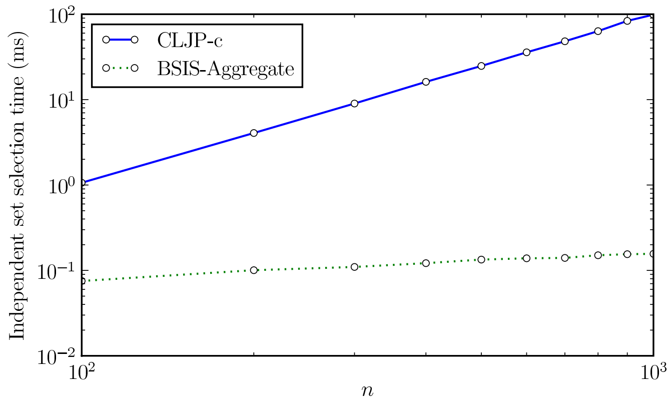

Figure 4.8: Total time for selecting independent sets in CLJP-c and BSIS with aggregate weight updates for the rotated anisotropic diffusion problem (4.2). The number of unknowns per trial is n^2.

4.5. Parallelizing BSIS. In this section, we discuss the prospects for developing a parallel BSIS algorithm. We first discuss coarsening for shared-memory architectures and then explore BSIS for distributed-memory computers.

Shared-memory parallelism is important to consider, since on-chip parallelism is now the primary mechanism for scaling processor performance as transistor sizes decrease.

For the search phase, CLJP-c benefits greatly from the presence of many threads, due to its many local tasks. The amount of parallelism in each iteration of BSIS depends on the number of vertices in the eliminated bucket, which is often few. This means that there is less parallelism for BSIS to exploit in each iteration, but this is not a weakness because BSIS has optimized the search phase to such a point that it is almost cost-free.

Both BSIS and CLJP-c benefit from multiple threads while computing updates, although BSIS with aggregate updates has the advantage that it is able to proceed despite knowing the weight of a vertex is to be updated. This means that weight update in this scheme could be overlapped with search.

The setup phase of BSIS takes more time for similar problems than the setup phase for CLJP-c, and this is due to the cost of loading the bucket data structure. Exploiting parallelism here will widen the gap between BSIS and CLJP-c.

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

For this discussion, assume that $G(S)$ is decomposed into p disjoint partitions, such that each $i \in V$ belongs to exactly one partition. Furthermore, assume that each partition is mapped to a processing core. In this model, computation is done on each partition, but dependencies between vertices on different partitions necessitate communication between partitions. Minimizing the frequency and the amount of communication is a goal in achieving scalability.

Our parallel BSIS algorithm is called the painted boundary method, and takes advantage of the invariance between coarse grids selected by BSIS and CLJP-c. We propose using BSIS to select as many of the interior C-points as possible prior to any communication with

ETNA
Kent State University
http://etna.math.kent.edu

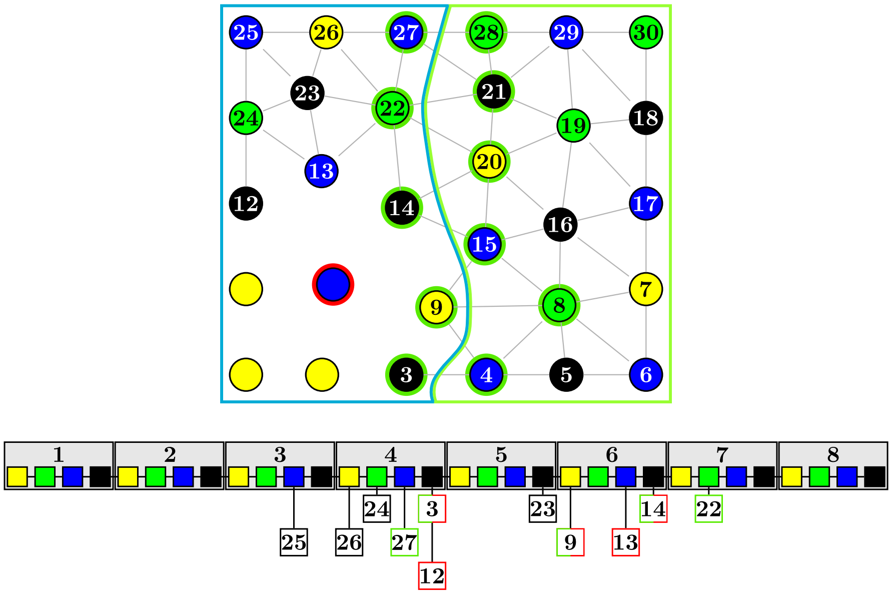

Figure 4.9: Painted boundary method with aggregate weight updates following one iteration. The data structure is for the left partition. Painted vertices are marked with a green ring in the graph and a green box in the data structure. Vertices in the data structure that are half green and half red are painted and also have had their weights updated.

neighboring partitions. All vertices belonging to a partition are colored and inserted into the BSIS data structure. The partition boundary vertices, however, are “painted”. If a painted vertex is in a set D in some iteration, then the vertex is not added to C. It is instead removed from the data structure and its neighbors in the same partition are also painted. Figure 4.9 illustrates the first iteration of the painted boundary method with weight update aggregation. The data structure shown is for the left partition.

The first iteration selects a new C-point, but does not select any painted vertices. In the second iteration, vertex 22 is selected, but is already painted. Therefore, same-partition neighbors of vertex 22 are also painted; see Figure 4.10.

The method finishes when the data structure is emptied of all vertices. The result is now three disjoint sets: C and F, as usual, but also a set of painted vertices. The painted vertices are the vertices that cannot be assigned to the F or C set without communication between partitions. The information is provided to CLJP-c, which handles the parallel portion of the algorithm.

The painted boundary approach is ideal when large numbers of interior vertices are given, which can be guaranteed on most problems for the fine grid. A side-effect of H1-based coarsening algorithms, however, is the creation of denser graphs on coarse levels. The issue is less of a concern for H1′-based coarsening using BSIS, wherein operator complexities are smaller and less dependent on problem size. In all cases, however, the number of processor boundary vertices relative to the number of interior vertices increases as the number of unknowns per processor decreases; e.g., on coarse levels. A few techniques may be applicable in this situation. The easiest solution is to simply use CLJP-c, or some other coarsening algorithm, on the coarser levels where few vertices are found. A second approach is the application of a dynamic load-balancing algorithm to decrease communication and possibly decrease the

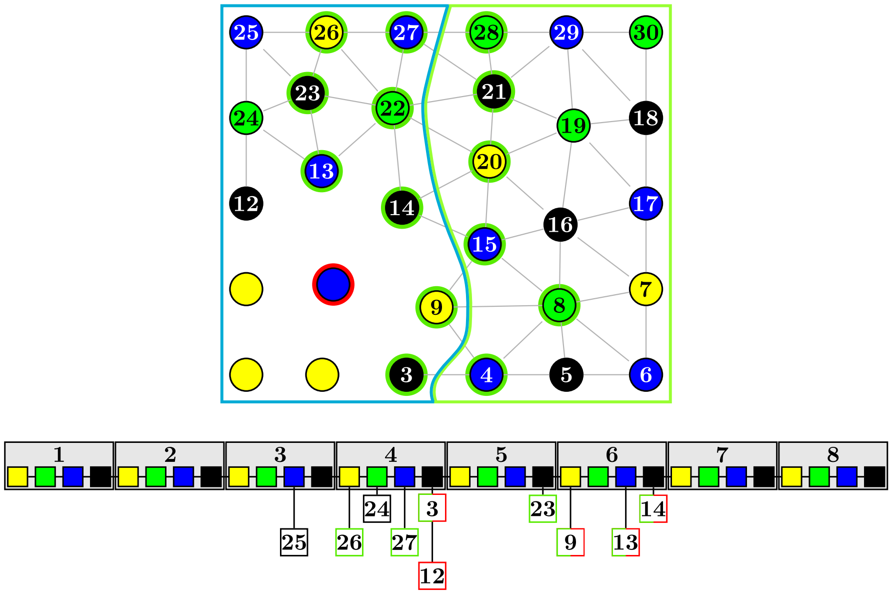

Figure 4.10: Painted boundary method with aggregate weight updates following the second iteration. In this iteration a painted vertex was selected, leading to the painting of its same-partition neighbors.

number of partitions on coarse levels. If the number of vertices per partition per level is maintained at a high enough level, BSIS is still valuable on coarse grids. A third option is to replicate the operator matrix on compute nodes, which leads to processors doing some identical work, but by avoiding communication. The second and third ideas are similar in nature and both involve using dynamic load-balancing techniques [11, 13, 15–17, 26].

5. Conclusions. In this paper, we have made two main contributions. First, new theoretical results were presented to expose the nature of independent-set based coarse-grid selection. Algorithms using the same rules, with some restrictions, for weight initialization, independent set selection, and weight update produce the same coarse grids, regardless of the particular implementation of the rules. This result decouples the design of policies for selecting quality coarse grids from the design and efficiency of the algorithms used to select the coarse grids. Our second contribution builds on these theoretical results with a new coarsening algorithm called BSIS, which selects the same coarse grids as CLJP-c, but uses a more efficient algorithm to select independent sets.

REFERENCES

[1] D. ALBER, *Modifying CLJP to select grid hierarchies with lower operator complexities and better performance*, Numer. Linear Algebra Appl., 13 (2006), pp. 87–104.
[2] D. ALBER AND L. OLSON, *Parallel coarse-grid selection*, Numer. Linear Algebra Appl., 14 (2007), pp. 611–643.
[3] W. N. BELL, L. N. OLSON, AND J. SCHRODER, *Pyamg: Algebraic Multigrid Solvers in Python. Version 1.0*, 2008. Available at http://www.pyamg.org.
[4] A. BRANDT, *General highly accurate algebraic coarsening*, Electron. Trans. Numer. Anal., 10 (2000), pp. 1–20. http://etna.math.kent.edu/vol.10.2000/pp1-20.dir/.

[5] A. Brandt, S. McCormick, and J. Ruge, *Algebraic multigrid (AMG) for sparse matrix equations*, in Sparsity and Its Applications, D. J. Evans, ed., Cambridge University Press, Cambridge, U.K., 1984, pp. 257–284.

[6] J. Brannick, *Adaptive Algebraic Multigrid Coarsening Strategies*, Ph.D. thesis, Department of Applied Mathematics, University of Colorado at Boulder, 2005.

[7] J. Brannick, M. Brezina, S. MacLachlan, T. Manteuffel, and S. McCormick, *An energy-based AMG coarsening strategy*, Numer. Linear Algebra Appl., 13 (2006), pp. 133–148.

[8] J. J. Brannick and R. D. Falgout, *Compatible relaxation and coarsening in algebraic multigrid*, SIAM J. Sci. Comput., 32 (2010), pp. 1393–1416.

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

[10] J. S. Butler, *Improving coarsening and interpolation for algebraic multigrid*, Master's thesis, Department of Applied Mathematics, University of Waterloo, 2006.

[11] U. V. Catalyurek, E. G. Boman, K. D. Devine, D. Bozdag, R. T. Heaphy, and L. A. Riesen, *Hypergraph-based dynamic load balancing for adaptive scientific computations*, in Proc. of 21st International Parallel and Distributed Processing Symposium (IPDPS'07), IEEE Computer Society, Los Alamitos, USA, 2007, p. 68.

[12] A. Cleary, R. Falgout, V. Henson, and J. Jones, *Coarse-grid selection for parallel algebraic multigrid*, in Solving Irregularly Structured Problems in Parallel, A. Ferreira, J. Rolim, H. Simon, and S.-H. Teng, eds., Lecture Notes in Computer Science, 1457, Springer, Heidelberg, 1998, pp. 104–115. 10.1007/BFb0018531.

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

[14] H. de Sterck, U. M. Yang, and J. J. Heys, *Reducing complexity in parallel algebraic multigrid preconditioners*, SIAM J. Matrix Anal. Appl., 27 (2006), pp. 1019–1039.

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

[16] K. D. Devine, E. G. Boman, R. T. Heaphy, R. H. Bisseling, and U. V. Catalyurek, *Parallel hypergraph partitioning for scientific computing*, in Proc. of 20th International Parallel and Distributed Processing Symposium (IPDPS'06), IEEE Computer Society, Los Alamitos, USA, 2006, p. 102.

[17] K. D. Devine, E. G. Boman, and G. Karypis, *Partitioning and load balancing for emerging parallel applications and architectures*, in Parallel Processing for Scientific Computing, M. Heroux, A. Raghavan, and H. Simon, eds., SIAM, Philadelphia, 2006, pp. 99–126.

[18] M. Griebel, B. Metsch, D. Oeltz, and M. A. Schweitzer, *Coarse grid classification: a parallel coarsening scheme for algebraic multigrid methods*, Numer. Linear Algebra Appl., 13 (2006), pp. 193–214.

[19] V. E. Henson and U. M. Yang, *BoomerAMG: A parallel algebraic multigrid solver and preconditioner*, Appl. Numer. Math., 41 (2002), pp. 155–177.

[20] A. Krechel and K. Stüben, *Parallel algebraic multigrid based on subdomain blocking*, Parallel Comput., 27 (2001), pp. 1009–1031.

[21] O. E. Livne, *Coarsening by compatible relaxation*, Numer. Linear Algebra Appl., 11 (2004), pp. 205–227.

[22] S. MacLachlan and Y. Saad, *A greedy strategy for coarse-grid selection*, SIAM J. Sci. Comput., 29 (2007), pp. 1825–1853.

[23] L. N. Olson, J. Schroder, and R. S. Tuminaro, *A new perspective on strength measures in algebraic multigrid*, Numer. Linear Algebra Appl., 17 (2009), pp. 713–733.

[24] J. W. Ruge and K. Stüben, *Algebraic multigrid (AMG)*, in Multigrid Methods, S. F. McCormick, ed., Frontiers in Applied Mathematics, 3, SIAM, Philadelphia, 1987, pp. 73–130.

[25] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, 2003.

[26] K. Schloegel, G. Karypis, and V. Kumar, *A unified algorithm for load-balancing adaptive scientific simulations*, in Supercomputing '00: Proceedings of the 2000 ACM/IEEE conference on Supercomputing, IEEE Computer Society, Los Alamitos, USA, 2000, p. 59.

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

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